ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

INGENIERÍA TÉCNICA EN INFORMÁTICA DE SISTEMAS

UN MEDIDOR DE RENDIMIENTO DE SERVIDORES DE BASES DE DATOS RELACIONALES

Realizado por

JOSE ANTONIO JAMILENA DAZA

Dirigido por

ANTONIO CÉSAR GÓMEZ LORA

Departamento

LENGUAJES Y CIENCIAS DE LA COMPUTACIÓN

UNIVERSIDAD DE MÁLAGA

MÁLAGA, SEPTIEMBRE DE 2009

UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

INGENIERÍA TÉCNICA EN INFORMÁTICA DE SISTEMAS

Reunido el tribunal examinador en el día de la fecha, constituido por:

Presidente/a Dº/Dª. ____

Secretario/a Dº/Dª.____

Vocal Dº/Dª. ____

para juzgar el proyecto Fin de Carrera titulado:

UN MEDIDOR DE RENDIMIENTO DE SERVIDORES DE BASES DE DATOS RELACIONALES.

del alumno Dº. Jose Antonio Jamilena Daza

dirigido por Dº. Antonio César Gómez Lora

ACORDÓ POR ___________________________________

OTORGAR LA CALIFICACIÓN DE ____

Y PARA QUE CONSTE, SE EXTIENDE FIRMADA POR LOS

COMPARECIENTES DEL TRIBUNAL, LA PRESENTE DILIGENCIA.

Málaga, a _____ de _____________ del 200___

El/La Presidente/a	El/La Secretario/a	El/La Vocal
Fdo:	Fdo:	Fdo:

Un medidor de rendimiento de servidores de bases de datos relacionales

Realizado por Jose Antonio Jamilena Daza

Dirigido por Antonio César Gómez Lora

CONTENIDO

TABLA DE ILUSTRACIONES

1. INTRODUCCIÓN

La mayoría de los sistemas gestores de bases de datos actuales disponen de herramientas propias o de terceros destinadas a medir el estado de salud y el rendimiento del propio sistema. Estos sistemas con capaces de proporcionar un extenso y valioso conjunto de información para administradores y auditores. Adicionalmente existen plataformas software destinadas a construir medidores de calidad de servicio, aunque, hoy por hoy, con una orientación muy específica hacia redes y servicios web.

Con este rico abanico de herramientas disponibles, y la necesidad de medir la calidad de los servicios prestados, es cada vez más común encontrar servidores dentro de una empresa u organización destinados a analizar la disponibilidad y rendimiento de los servicios ofertados (entendiendo el acceso a una base de datos como uno de estos servicios). Desgraciadamente estos servidores suelen estar situados estratégicamente en zonas clave de las intranets o de las infraestructuras de comunicación. Cuando los servicios están abiertos al exterior sólo se analiza el entorno próximo del servidor y nunca el entorno del cliente.

Para un usuario simple que accede desde su equipo a un servidor de bases de datos el rendimiento viene marcado por el rendimiento de todos y cada uno de los equipos, dispositivos y enlaces involucrados en el procesamiento y el transporte de la información. En una sesión común esto puede significar la participación de decenas, cientos e incluso miles de elementos hardware y software participantes, la mayoría de los cuales son transparentes al usuario. Y en caso de ser visible alguno de estos participantes generalmente no se tiene la capacidad ni los permisos suficientes para analizar su rendimiento individual.

Se nos plantea pues la necesidad de que un usuario pudiera ejecutar un sencillo programa que le permitiera medir el rendimiento de determinadas tareas en un servidor de bases de datos. Este sistema analizará el rendimiento de estas tareas de forma

periódica siguiendo una planificación dada para cumplir su objetivo principal, que es detectar la aparición de posibles problemas. Aunque el diagnóstico del problema queda fuera de los objetivos de esta aplicación, sí es cierto que se añadirá la capacidad de clasificar las diferentes tareas según características particulares para agrupar sus resultados y así asistir al usuario en la interpretación de los datos y estadísticos para que se pueda realizar un diagnóstico. Este programa sencillo tiene un doble destino: primero el ser utilizado por usuarios aislados; y segundo el poder ser instalado en múltiples equipos y permitir recolectar posteriormente sus datos de forma sencilla (preferiblemente en forma de un archivo simple).

1.1. OBJETIVOS

Los objetivos que este proyecto plantea son los que siguen:

- Elaborar un software que ejecute sobre sistemas gestores de bases de datos relacionales, bajo una planificación dada, unos scripts de prueba, para a partir de los tiempos de ejecución generar indicativos sobre su comportamiento.

- Construir una interfaz gráfica que sea capaz de recoger los datos de la herramienta descrita anteriormente y generar unas estadísticas que sean mostradas de forma visual para que el usuario las interprete. Dichas estadísticas se agruparan según los diferentes tipos de scripts lanzados.

- Generar un conjunto base de scripts que puedan ser clasificados en conjuntos no disjuntos de categorías o propiedades. Con esto se consiguen estadísticas según las posibles características evaluar.

1.2. METODOLOGÍAS Y FASES DE TRABAJO

En el desarrollo del proyecto se empleará el paradigma de programación orientado a objetos y para modelar el software el Lenguaje Unificado de Modelado o UML, iniciales que corresponden al acrónimo inglés *Unified Modeling Language*.

El modelo de desarrollo será el Desarrollo Iterativo Incremental [1].

En cuanto a las fases en las que se dividirá el desarrollo software éstas serán:

- Una prospección inicial del desarrollo en el que barajaremos posibles lenguajes para desarrollar el proyecto, tecnologías a emplear, etc. Así también como la obtención de los objetivos que ha de cumplir el desarrollo software que formará el proyecto. Tras la cual habremos obtenido el análisis de requisitos.

- Se realizará la especificación de los distintos componentes que formarán parte del desarrollo software.

- Una vez que tenemos los componentes que debemos implementar pasaremos a una fase de diseño de la arquitectura de la solución. En esta fase del desarrollo se realizará el diseño de los distintos componentes del sistema software, así como el diseño de la jerarquía de clases de los distintos componentes del desarrollo software.

- Posteriormente pasaremos a la implementación del diseño. En esta fase se puede llegar a tener que hacer cambios en el diseño si la naturaleza del desarrollo así lo exige.

- Finalmente se realizarán las pruebas oportunas para verificar el correcto comportamiento del sistema.

Durante todo el proceso se seguirá el Desarrollo Iterativo Incremental [1], se irán obteniendo prototipos de desarrollo sobre los que se irán haciendo pruebas de funcionamiento y de completitud de requisitos.

Durante todas estas fases se llevará un amplio proceso de documentación, tanto para la escritura de la memoria final como para guardar el conocimiento adquirido durante el desarrollo.

1.3. HERRAMIENTAS DE DESARROLLO

El desarrollo se hará sobre el lenguaje de programación Java, concretamente haciendo uso de JDK 1.6.0_14-b08 de Sun Microsystems, la última versión de este leguaje de programación orientado a objetos [2].

El interfaz elegido para el desarrollo será NetBeans 6.5, un entorno integrado de desarrollo especializado en Java [3].

Todo el proyecto se guardará con el sistema de control de versiones Subversion. Este se encarga de mantener históricos de ficheros durante los desarrollos software [4]. Subversión se instala como un software servidor al que se accede mediante un cliente. El cliente elegido es TortoiseSVN, un cliente que se integra tanto en la shell del sistema operativo como en el propio NetBeans [5].

Se realizará el diseño de la aplicación sobre Enterprise Architect y sobre un complemento específico de NetBeans para estos menesteres. Enterprise Architect es una herramienta CASE, Computer Aided Software Engineering o Ingeniería de Software Asistida por Ordenador, y en el diseño de este proyecto en cuestión se guardarán los datos de diseño en un sistema de bases de datos MySQL [6][7]. En cuanto al componente de NetBeans lo almacenará con estructura de archivos de proyecto de NetBeans.

El proyecto tendrá integrada su propia base de datos de estadísticas. Para dicho cometido se empleará SQLite, un sistema gestor de bases de datos empotrado, rápido y de poco peso en memoria [8]. Para comunicarse con Java se empleará la librería SQLiteJDBC [9]. Para facilitar el diseño en dicho sistema gestor de bases de datos se empleará la herramienta SQLite Management Studio, una herramienta para gestión gráfica de ficheros SQLite [10].

Para velar por la integridad y conocer inequívocamente los fallos que pueden acaecer, tanto en la ejecución normal como en las sesiones de pruebas, se empleará Apache Log4J. Apache Log4J es un sistema de diario de ejecución para aplicaciones Java [11]. Dicho sistema nos da la capacidad de generar ficheros de diario de ejecución con las trazas de la misma e incluso la capacidad de que llegado el caso, si el sistema al que estamos evaluando se llegara a colapsar, enviar un email advirtiendo este hecho.

Para el componente encargado de mostrar las estadísticas de las pruebas se empleara un interfaz gráfico Java implementado con Swing. Como sistema visualizador de datos y estadísticas emplearemos la biblioteca JFreeChart. JFreeChart es un sistema de generación de graficas para el entorno grafico Swing de Java [12].

Las conexiones entre el proyecto y los diferentes gestores de bases de datos se harán mediante la tecnología JDBC (Java Data Base Connectivity). JDBC es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java, independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede, utilizando el dialecto SQL del modelo de base de datos que se utilice.

Toda la planificación temporal del proyecto será llevada acabo con la ayuda de OpenProj [13]. Y la documentación se generará a partir de Enterprise Architect y se editará el texto con OpenOffice [14] y con Microsoft Visio [15] los diagramas que fueran necesarios.

1.4. CALIDAD

La calidad debería ser un principio fundamental en el desarrollo software, y en especial si el desarrollo lo estamos haciendo en un lenguaje como Java que dispone de multitud de estándares y herramientas de auditoría que nos permiten garantizar que el código implementado cumple todos los requisitos como para considerarlo estable y seguro.

Para la completitud de los propósitos de calidad el desarrollo se ha empleado herramientas de verificación de código estático y de chequeo de código.

Para la verificación de código estático se ha empleado PMD. PMD analiza el código fuente de Java y busca posibles problemas como:

- Posibles errores: sentencias try / catch / finally / switch vacías.
- Código de muertos: variables locales, parámetros y métodos privador sin usar.
- Optimiza código: controla el despilfarro de uso de String / StringBuffer.
- Detecta expresiones excesivamente complejas, declaraciones innecesaria o bucles sin condiciones de salida.
- Detecta código duplicado. El copiar / pegar en el código significa copiar / pegar los errores.

Para garantizar que la escritura de código es estándar y portable se emplea Checkstyle. Checkstyle es una herramienta de desarrollo para ayudar a los programadores escribir código Java que se adapte a una norma de codificación.

Automatiza el proceso de verificación y control de código Java ya que para los seres humanos de esto, a pesar de ser una tarea importante, resulta muy aburrido.

Esto hace que sea ideal para los proyectos que quieren hacer cumplir una norma de codificación.

Checkstyle es altamente configurable y se puede hacer para apoyar casi cualquier norma de codificación. Por defecto este proyecto final de carrera emplea los convenios de Sun para el desarrollo de código Java.

1.5. CAPÍTULOS

La memoria consta de x capítulos en los que se abordan los siguientes temas en cada uno de ellos.

1. En el primer capítulo nos encontramos la introducción.

2. En el segundo capítulo analizamos el mercado de este tipo de soluciones y comparamos con la llevada a cabo en este proyecto final de carrera.

3. En el tercer capítulo, nos dedicamos a abordar los distintos tipos de soluciones por las que podía haber abordado el problema, plantearemos sus ventajas e inconvenientes, y nos decantaremos por una que es la que llevaremos a su culminación.

4. En el cuarto capítulo nos embarcamos en las distintas herramientas o bibliotecas para el desarrollo y la gestión del proyecto. Indagaremos sobre cuales vamos a emplear y el porqué.

5. En el quinto capítulo empezamos con el diseño de la solución.

6. El sexto se tratará algo de calidad de software aplicada al desarrollo del proyecto final de carrera.

7. Conclusiones y línea futura.

2. PROSPECCIÓN EN EL MERCADO DE PRODUCTOS QUE ABORDEN EL PROBLEMA TRATADO

El análisis y la métrica de rendimiento en los sistemas gestores de bases de datos relacionales es una práctica muy extendida. Actualmente todos los sistemas gestores de bases de datos, de una forma u otra, disponen de herramientas que ayudan tanto a su configuración como a la monitorización y supervisión de los mismos. Entonces, ¿por qué diseña otro medidor de rendimiento?

La explicación es bien sencilla, y la existencia de sistemas de métrica y supervisión de sistemas gestores de bases de datos de terceros nos dan la razón. Normalmente, las herramientas que se incluyen en con el propio sistema gestor de bases de datos están orientadas a analizar parámetros propios de rendimiento, de estado del servicio dentro del sistema operativo, de la apertura o no de los canales de comunicación por los que el sistema gestor de bases de datos se comunica con los clientes ya sean usuarios o software de terceros. Estas métricas son concienzudas y han sido realizadas por los propios creadores del gestor de bases de datos, y nadie mejor que ellos para analizar en modo teórico todo lo relacionado con dicho gestor.

Pero, ¿qué ocurre cuando por ejemplo, aquí en la Escuela, en un examen de la asignatura de bases de datos trescientos alumnos se enfrentan a un examen en el que todos de modo concurrente tienen que lanzar consultas contra un determinado gestor de bases de datos y todo va mal? Sí se consultan las herramientas de supervisión y se encuentra algún problema, es fácil de solucionar. Pero ¿y si esto no ocurre? ¿De dónde viene el fallo? ¿Es fallo del sistema gestor de bases de datos y las herramientas de supervisión no lo reflejan o por lo menos de un modo claro? ¿Puede ser causado por la red? ¿Puede ser del sistema operativo del servidor que está actualizando algún componente, verbigracia el parseador de XML y el planificador de sistema operativo lo

considera de más prioridad que el servicio del sistema gestor de bases de datos? ¿Puede ser un alumno rezagado que no ha estudiado las sentencias JOIN y ha creado sin saberlo un producto cartesiano inmenso que desborda la caché y los tiempos de CPU del gestor de bases de datos?

Con este proyecto final de carrera no se pretende hacer una herramienta que prevenga cualquier tipo de fallos y los solucione. Lo que se pretende desarrollar es un sistema muy simple para medir rendimientos, que se pueda ejecutar de una forma eficiente y casi imperceptible para el usuario y que nos dé una idea de que algo anormal está ocurriendo.

En el marcado ya existe herramientas con estas capacidades, otras que desarrollan funciones similares.

2.1. Transaction Processing Performance Council (TPC)

TCP es una organización sin fines de lucro fundada en 1988 para definir los proceso de transacciones y herramientas de métrica de base de datos además de difundir los resultado de la información de forma objetiva y verificable para la industria.

Los sistemas de métrica de TPC son ampliamente utilizados hoy en la evaluación del rendimiento de los sistemas, los resultados se publican en el sitio web de TPC.

Estos sistemas de métrica son tipo benchmark y están más orientados a la comparación de distintos sistemas gestores de bases de datos.

No los podemos incluir en el mismo saco que este proyecto, ya que los objetivos pretendidos son distintos, pero fue la primera toma de contacto con este tipo de

soluciones. Además de disponer de múltiple documentación de cómo hacer pruebas de rendimiento para poder encontrar posibles fallos de rendimiento o de degradación en el comportamiento normal del sistema. También emplean metodologías de desarrollo de sus benchmark de forma altamente eficiente.

2.2. NetIQ AppManager for Oracle Database RDBMS Server

Dispone de un sistema de administración y supervisión de rendimiento. Proporciona herramientas de gestión de bases de datos Oracle optimizando su rendimiento y disponibilidad. Disponible en tiempo real de sistemas de diagnóstico y se almacenan en un repositorio de históricos.

Proporciona un sistema de notificación proactiva y un sistema automático que aplica medidas correctivas. Además de un sistema de aviso de todas las ocurrencias que se producen.

Aunque orientado para gestores de bases de datos Oracle. A pesar de todo, también se pueden adquirir componentes para hacerlo trabajar con Microsoft SQL Server.

Es un software propietario. Y a diferencia de los anteriores no dispone de ningún conjunto de bibliotecas para el desarrollo de complementos.

Sólo se puede implantar en entornos Microsoft Windows.

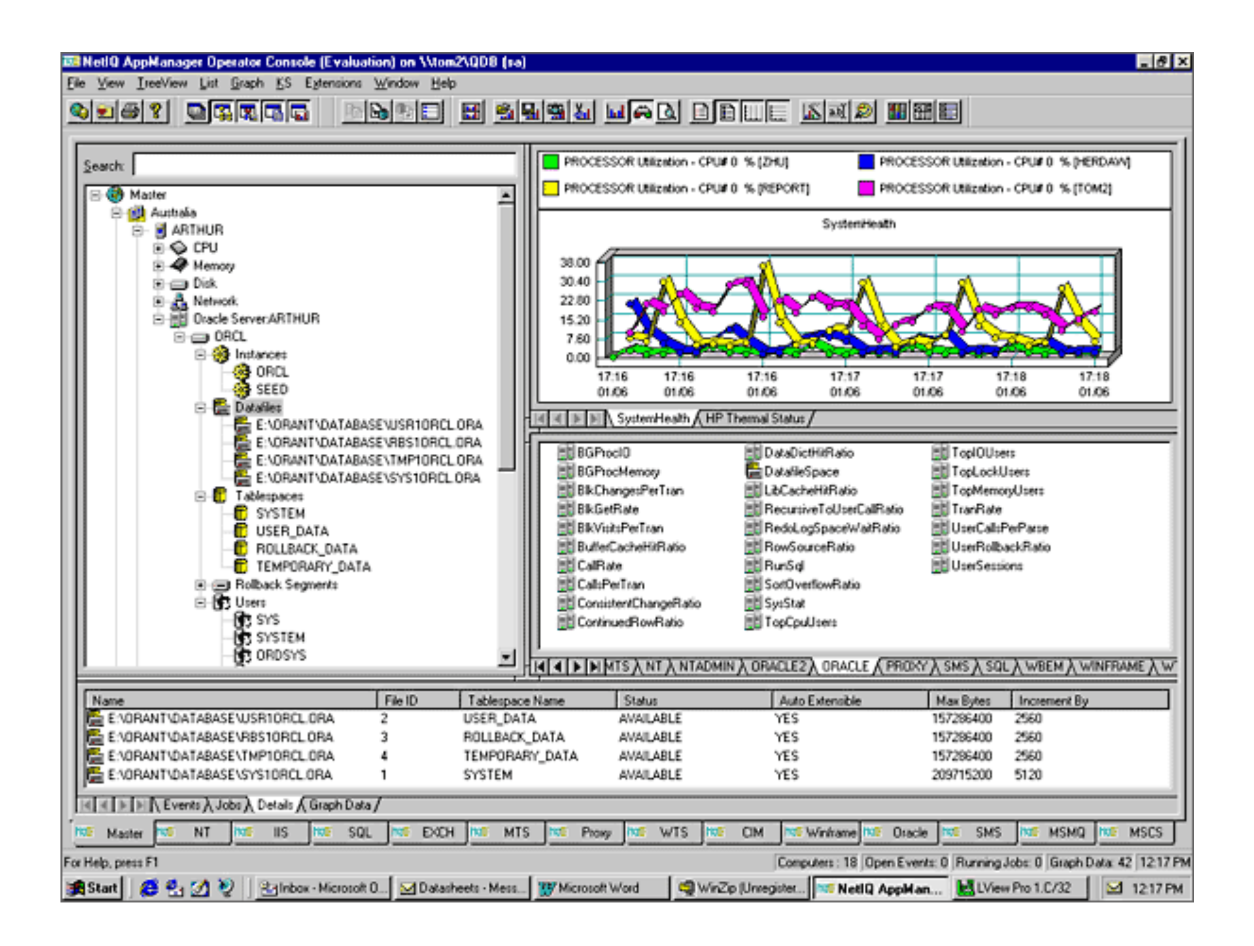


Ilustración 1 – Ilustración obtenida de www.netiq.com

2.3. NAGIOS

Nagios es un sistema de código abierto para la supervisión de redes. Nagios está muy extendido. Nagios se emplea para la supervisión de equipos y servicios.

Nagios se distribuye bajo licencia GNU General Public License Version 2.

Es un software que proporciona una gran variedad de parámetros a consultar y supervisar en el sistema y genera alertas según pautas preestablecidas.

Entre sus virtudes figura la capacidad de supervisar protocolos de red tales como SMTP, POP3, HTTP y SNMP entre otros, la supervisión del hardware de los equipos

conectados en la red y dispone de un conjunto de bibliotecas para el desarrollo de complementos para el sistema.

El diseño de dichos componentes o plugins se puede hacer mediante Bash, C++, Perl, Ruby, Python, PHP, C#,...

En cuanto a su implantación, como aplicación es bastante pesada, por lo cual normalmente necesita un host dedicado a este fin.

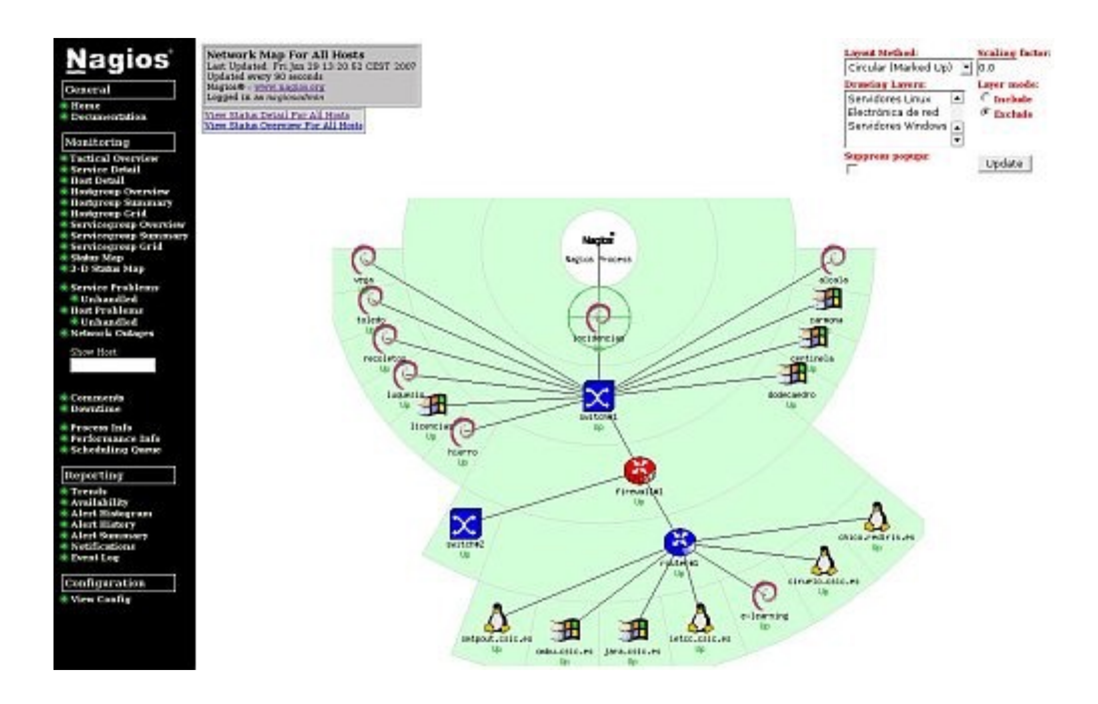


Ilustración 2 – Ilustración obtenida de www.nagios.org/

2.4. WEBMIN

Webmin un sistema de administración remota mediante HTTP para sistemas UNIX. Webmin permite configurar aspectos del sistema desde un navegador.

Webmin se distribuye bajo Licencia BSD.

Webmin está escrito en Perl, versión 5, y tiene su propio servidor HTTP ligero y con poco consumo de recursos.

Webmin es un sistema modular, y el usuario puede decidir que quiere y que no quiere cargar en la administración. Dispone a sí mismo un conjunto de bibliotecas para el desarrollo de estos módulos.

No está orientado al desarrollo de módulos no orientados a administración de sistemas.

Se implanta en sistemas UNIX, aunque existen proyecto para portarlo a Windows estas no son oficiales.

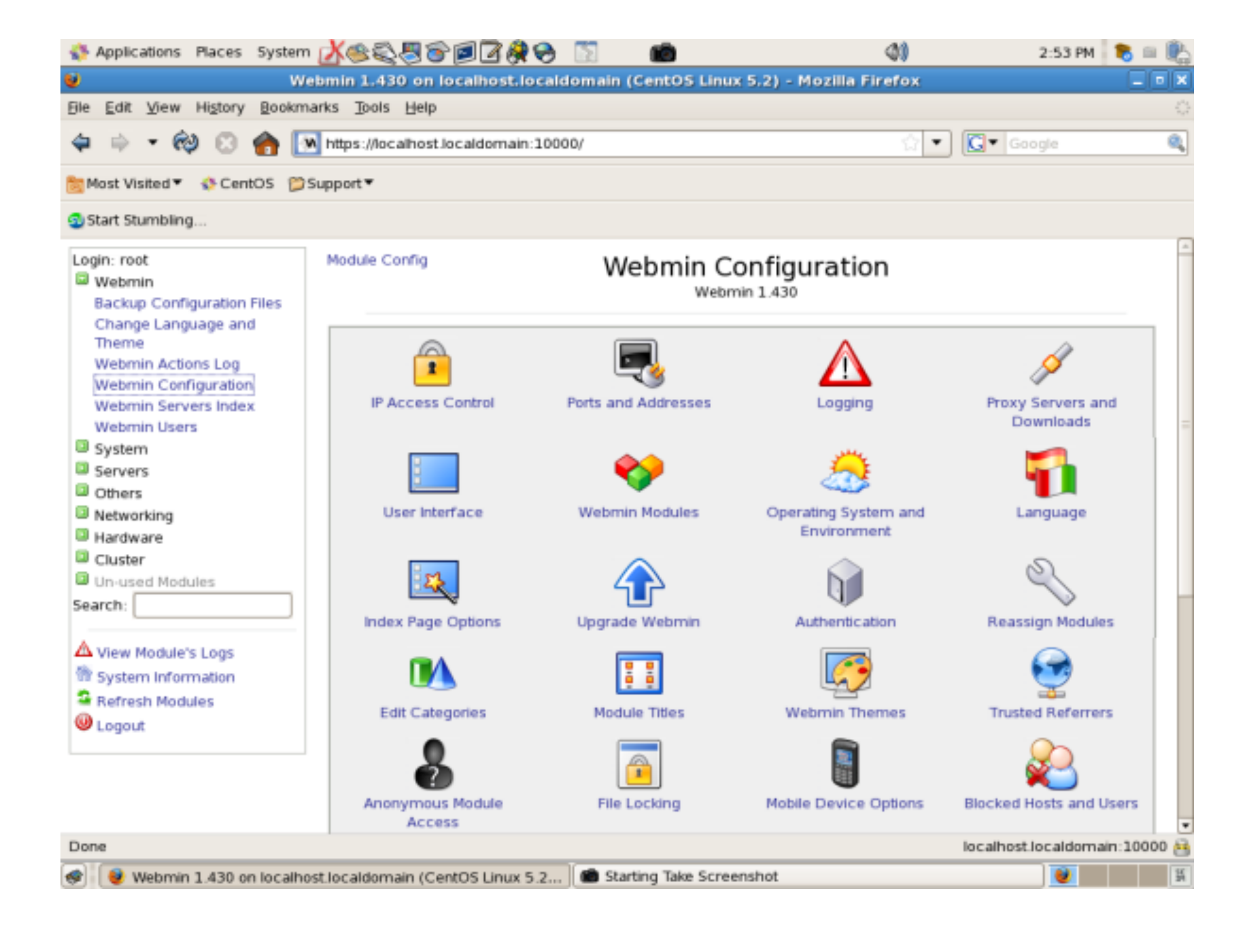


Ilustración 3 – Ilustración obtenida de www.webmin.com

2.5. Ventajas e inconvenientes de este Proyecto Final de Carrera con respecto a los anteriores

Ahora pasamos a mostrar una tabla comparativa de los sistemas anteriormente descritos y la comparación de sus capacidades con las del proyecto final de carrera al que pertenece estas memorias.

La tabla comparativa no incluye los sistemas de supervisión de los distintos sistemas gestores de bases de datos, ya que estos pueden ser muy diferentes entre sí, y enumerarlas todas es una tarea ardua y como hemos comentado con anterioridad estas herramientas tiene capacidades superiores a las analizadas, ya que su ámbito de actuación es mucho más amplio y complejo.

	Nagios	**Webmin**	**NetIQ AppManager for Oracle Database RDBMS Server**	**PFC**
Conjunto de bibliotecas para desarrollo	SI	SI	NO	NO
Multiplataforma	NO (Host dedicado)	NO (Sistemas UNIX)	NO (Sistemas Windows)	SI (JVM)
Funcionalidad de propósito especifico	NO	NO	SI	SI
Implementado	NO	NO	SI	SI
Licencia	GPL v.2	BSD	Propietaria	Apache 2.0

Distribución	Gratuita	Gratuita	De pago	-
Generación de gráficas	-	-	SI	SI (Analizador)
Supervisión en tiempo real	-	-	SI	SI (Chartserver)
Servicio de sistema	NO (Host dedicado)	SI	SI	SI (Servicio)
Lenguaje nativo	C y otros	Perl	-	Java 1.6.0_14
Múltiples SGBD	-	-	NO (Oracle)	SI (JDBC)

Ilustración 4 – Comparativa entre aplicaciones

3. Distintos tipos de soluciones para el mismo problema

Para afrontar el desarrollo del proyecto final de carrera, lo primero fue intentar enfocar el problema desde diferentes perspectivas, analizando siempre sus ventajas e inconvenientes. En dichas comparaciones se ha tenido en cuenta las ventajas que podríamos obtener en el desarrollo del proyecto final de carrera, echando mano de distintos marcos de trabajo ya consolidados para crear la solución. Frente a los inconvenientes de crear una infraestructura completa para llevar a la consecución el proyecto.

También hemos de tener en cuenta las desventajas acaecidas del hecho de emplear dichos marcos de trabajo y los distintos problemas que puedan producirse por este hecho. A veces, si nos centramos mucho en intentar acatar una disciplina de trabajo producida por el uso de marco de trabajo determinado podemos perder el control del desarrollo y esto en un proyecto que tiene como baluarte la sencillez y eficiencia en la obtención de resultados puede pesar negativamente.

En los distintos enfoques para el problema se partió de la idea de no ligarse en exceso ni de una plataforma de ejecución, ni de ningún marco de trabajo determinado y cuyos cambios puedan influir negativamente en nuestro trabajo.

3.1. Componente para Nagios

Una de los planteamientos más estudiado fue esta. Nagios es un sistema estable y bastante extendido para las labores de métrica y de monitorización tanto de equipos como de servicios de red.

Nagios durante el tiempo en que se realizo la prospección del mismo no tenía ningún componente cuya finalidad fuera la misma que la que se pretende con este proyecto final de carrera, luego si se hubiera optado por este planteamiento, hubiera sido una propuesta nueva para este sistema.

Nagios además de la multitud de componentes que incluye, dispone de un potente conjunto de bibliotecas de desarrollo y que permiten la creación de componentes de cualquier tipo.

Tras el estudio que se llevo a cabo nos encontramos con inconvenientes que hacían no abordable este mecanismo para llevarlo a cabo. Dichos inconvenientes eran principalmente que Nagios necesita unos requerimientos hardware que prácticamente lo condicionan a ser desplegado en un host para uso exclusivo de dicha aplicación, y sobre todo si tenemos por ejemplo como escenario al que puede ir dirigido uno de los laboratorios del centro.

Aun así, pasemos a ver la arquitectura de la solución. El desarrollo de la solución sería un sistema cliente/servidor, siendo el servidor una maquina de uso exclusivo para Nagios. Los clientes un servicio instalado en las máquinas que deseen probar un determinado gestor de bases de datos. Obviamente, el propio servidor sería capaz de hacer pruebas sin necesidad de clientes, los clientes solo darían distintos puntos de vista del mismo escenario.

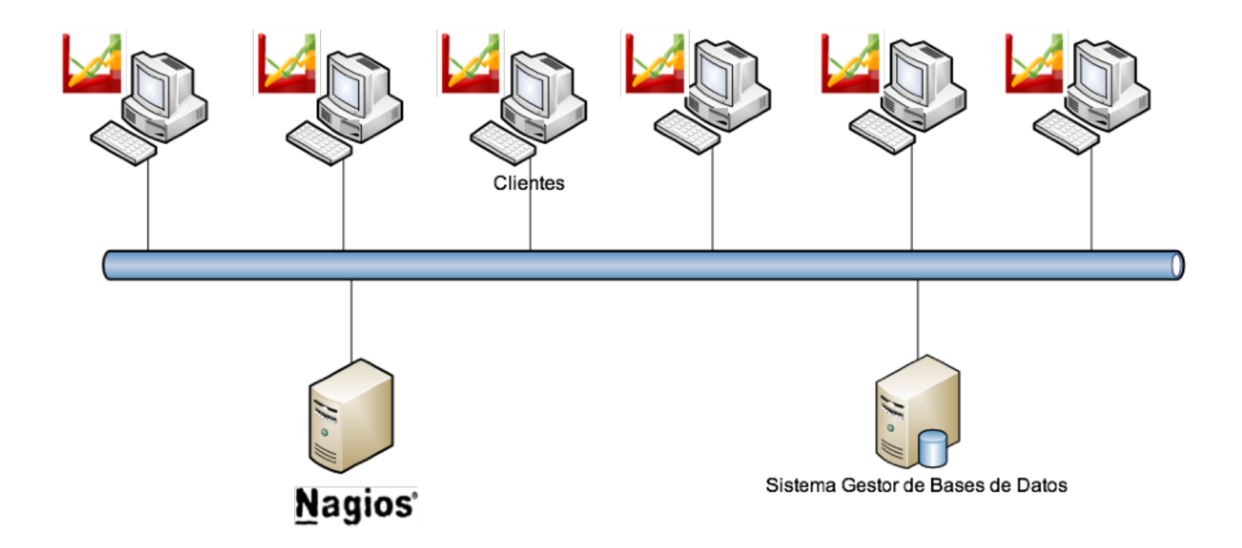


Ilustración 5 – Diagrama de solución con Nagios

La problemática de la inclusión a en nuestra infraestructura de red de un host dedicado a Nagios, suponiendo esto un gasto extra si nuestra red no dispone de ninguno, fue el factor principal de desecharlo.

3.2. Componente para Webmin

Siguiendo la prospección de distintos sistemas de monitorización y supervisión que se encuentran muy extendidos dentro de la administración de sistemas, abordamos el intento de desarrollo de otro componente, en este caso para Webmin.

Webmin es una plataforma de administración muy extendida. La opción de diseñar un complemento para dicho sistema que nos sirviera de solución para los propósitos perseguidos era bastante atrayente y más cuando comprobamos que no existía ninguno hecho que hiciera eso mismo.

Esta solución haría que cada equipo que tuviera el componente implantado, efectuaría peticiones sobre el sistema gestor de bases de datos y mostraría los resultados obtenidos. Lo que haría necesario que también se pudiera configurar el componente de

uno de los equipos como primario, y fuera el que recibiera toda la información de la granja de equipos que efectúan las pruebas.

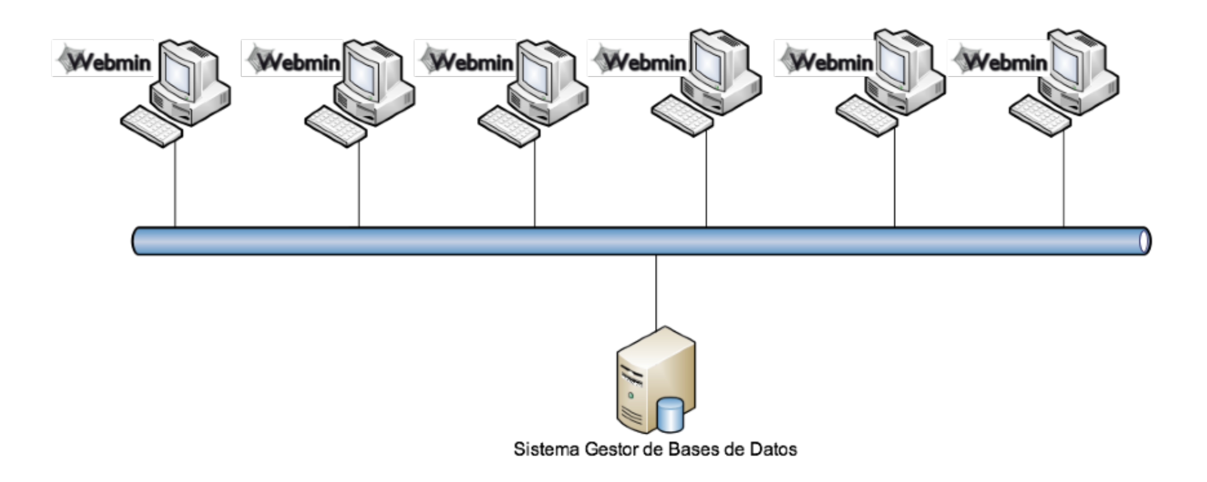


Ilustración 6 – Diagrama de solución con Webmin

Esta opción fue descartada por dos grandes motivos. Para empezar Webmin está orientado a correr sobre sistemas operativos tipo UNIX, lo que no nos garantiza la multiplataforma que era algo que se pretendía desde un principio, en realidad existen portaciones de Webmin para entornos Windows pero no son oficiales, ni tampoco se nos garantiza la fiabilidad y la compatibilidad de los mismos. El otro de los motivos es que las bibliotecas no tienen funciones que nos ayuden en nuestros propósitos, dichas bibliotecas están hechas en Perl lo que tampoco hacen demasiado atractivo el desarrollo que partiría prácticamente de cero y nos forzaría a hacerlas compatibles con todo los ya hecho, añadiendo un requisito extra externo al proyecto final de carrera.

Ambas condiciones mostraban un escenario hostil y del que no obtendríamos ninguna ventaja.

3.3. CLIENTE/SERVIDOR DESPLEGADO EN UN SERVIDOR DE APLICACIONES

La arquitectura constaría de un servidor de aplicaciones, como Glassfish, que desplegaría un Webservice que facilitaría la comunicación con los clientes que hagan las métricas y junto a una aplicación web que se encargaría de publicar la información estadística por web.

Otro posible planteamiento para dicha solución es cambiar el servidor de aplicaciones por un motor de integración[1], que aunque no esté muy relacionado con el problema, yo personalmente tengo mucho bagaje en estas herramientas y todas disponen de herramientas de desarrollo que nos ayudarían tanto a hacer las pruebas como en obtener los datos y la información derivada de los mismo.

El cliente lanzaría las consultas a tratar contra el sistema gestor de bases de datos y tomaría los tiempos obtenidos. Dichos tiempos se enviarían al Webservice que despliega el servidor. Éste se encargaría de procesar los datos recibidos y los publicaría sobre la aplicación web.

Dicho planteamiento tiene los mismos inconvenientes que el desarrollo de un componente para Nagios, requiere un host dedicado en este caso para el servidor de aplicaciones, por tanto tiene todos los problemas consecuentes de esto.

Por otro lado esta arquitectura, a nivel académico serviría para investigar un campo de conocimientos amplio como son los Webservice, el desarrollo de aplicaciones web, la comunicación entre plataformas y sistemas de diferente índole,…, pero como el objetivo de este proyecto final de carrera es conseguir algo eminentemente práctico y usable en un entorno real como pueda ser un laboratorio de la Escuela, se termino descartando esta solución.

---

[1] Un motor de integración es una herramienta de software diseñado para simplificar la creación y gestión de interfaces entre aplicaciones separadas y sistemas en una organización. Los motores de integración intercambian mensajes entre sistemas y permiten la gestión, mapeo, traducción y modificación de datos entre sistemas de información para asegurar el intercambio efectivo de datos en la organización.

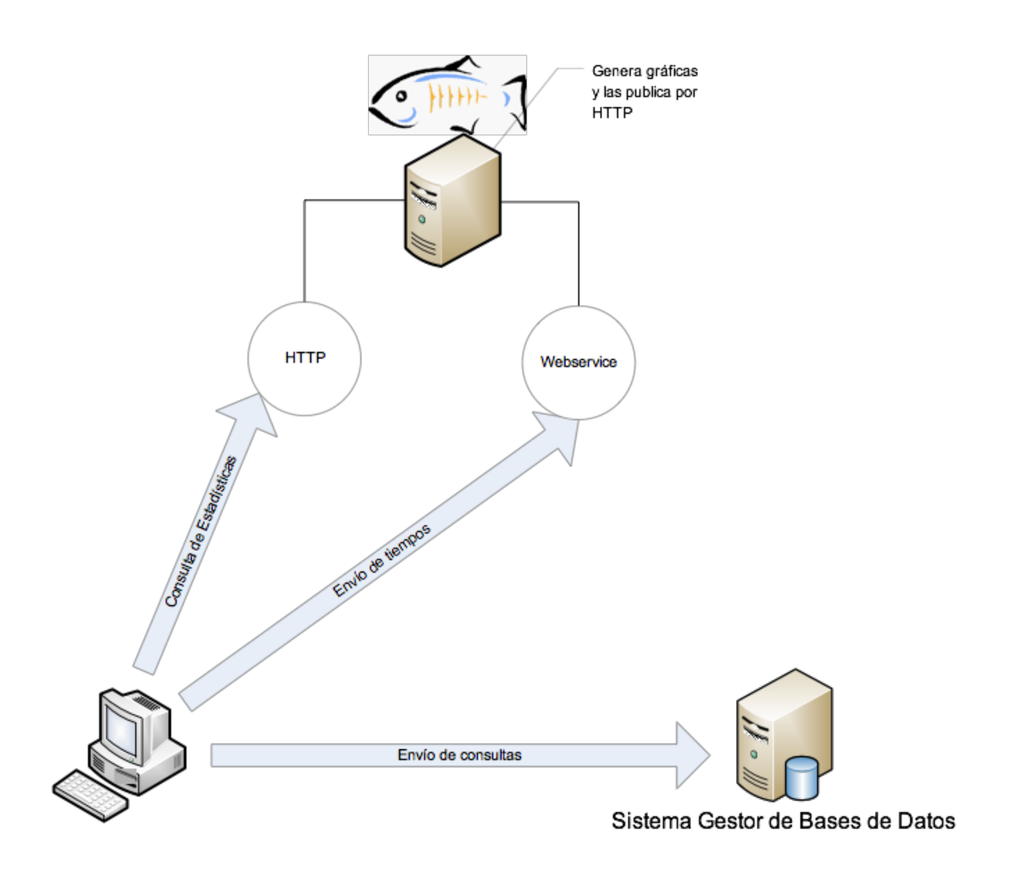


Ilustración 7 – Diagrama de solución mediante servidor de aplicaciones

3.4. SERVICIO/ANALIZADOR DE DATOS

Esta solución, que puedo adelantar que, ha sido la elegida para la consecución de este proyecto final de carrera.

En esta arquitectura se diferencian dos sistemas bien diferenciados. Uno un servicio, llamado curiosamente Servicio, que es el encargado de lanzar consultas y hacer las pertinentes métricas de tiempo. Y otro al que se le denomina Analizador y es el que no facilita la visión y análisis de los datos que proporciona el servicio mencionado anteriormente.

El servicio es un pequeño demonio de sistema integrable en el sistema operativo, con independencia de que este sea el que sea, y que efectuaría las peticiones al sistema gestor de bases de datos y guardaría los resultados es un fichero. En la práctica, el fichero en cuestión es una base de datos SQLite, aunque se puede configurar como un

acceso a otro sistema gestor de bases de datos, preferentemente no al que se le van a efectuar las pruebas aunque esto es factible.

El otro miembro de la dualidad sería el Analizador de los datos que se hayan guardado en el fichero de almacenamiento anteriormente indicado. Es un entrono gráfico con los pertinentes asistentes de interpretación.

Está arquitectura tiene como ventaja la simplicidad de desarrollo, no siendo esto por falta de trabajo o de calidad sino por su simpleza de conceptos y su buena diferenciación de campos de trabajo de cada elemento, ya que los dos componentes que la forman son independientes y su nivel de requisitos de ejecución es prácticamente despreciables.

Por otra parte, tiene la virtud de que un solo Analizador es capaz de interactuar con múltiples Servicios.

Por tanto, por ejemplo para medir la salud de un determinado sistema gestor de bases de datos desde un laboratorio de la Facultad, se puede lanzar un Servicio desde cada equipo de los alumnos del laboratorio, mientras el profesor puede consultar los resultados desde otro equipo usando una sola instancia del Analizador.

Dado el grado de sencillez del concepto de la solución y de su versatilidad, el desarrollo proyecto final de carrera se decanto por esta solución.

Para garantizar el concepto de multiplataforma, el desarrollo se hace sobre Java SE.

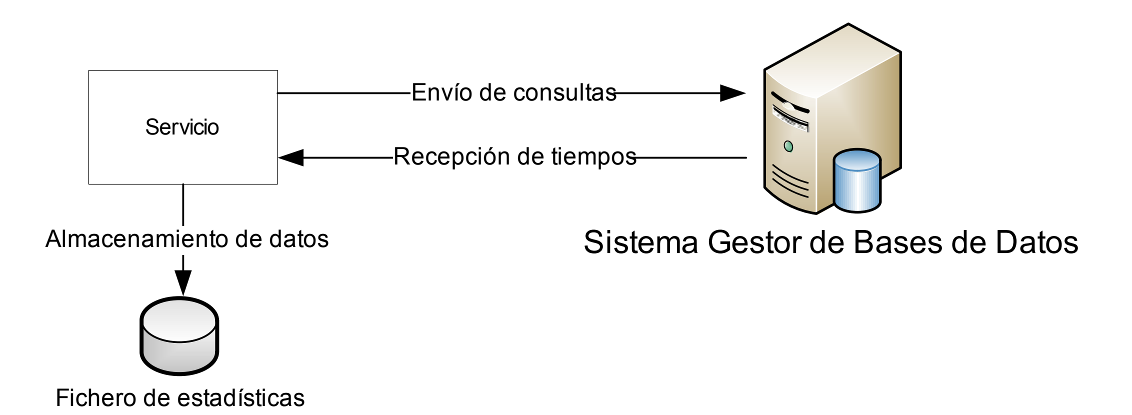


Ilustración 8 – Diagrama de aplicación Servicio

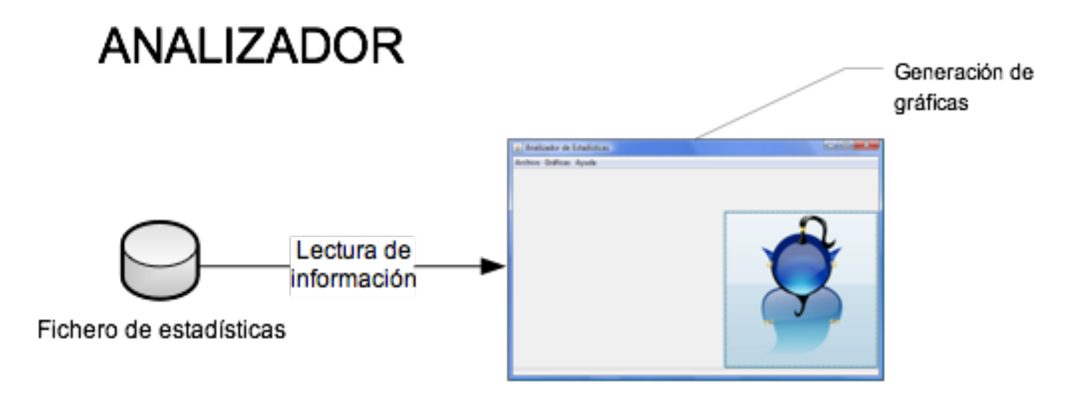


Ilustración 9 – Diagrama de aplicación Analizador

4. ARQUITECTURA Y HERRAMIENTAS EMPLEADAS PARA EL DESARROLLO Y LA GESTIÓN DEL PROYECTO.

En este capítulo comenzaremos la enumeración de los distintos elementos que ha participado en el desarrollo de este proyecto final de carrera.

4.1. LENGUAJE DE PROGRAMACIÓN: JAVA

Bueno para empezar hemos de tratar el asunto del lenguaje de programación. El lenguaje de programación elegido para el desarrollo ha sido Java. Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90.

El lenguaje se escribe con una sintaxis similar a C y C++, al que se le ha simplificado el modelo de objetos y elimina las sentencias de bajo nivel, ya que estas son las causantes de muchos errores, como la manipulación directa de punteros o memoria y que no suelen poder detectarse en tiempo de compilación. Java además dispone de sentencias de control de errores mediante el mecanismo de excepciones, minimizando las incidencias de los mismos.

Java se ejecuta sobre una máquina virtual (en inglés Java Virtual Machine, JVM) que es un programa nativo, es decir, ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en bytecode, una especie de ensamblador, el cual es generado por el compilador del lenguaje Java. La generación de los bytecode que son los mismos para cualquier implantación de la máquina virtual nos

garantiza el funcionamiento de nuestro código, siempre que la JVM cumpla correcta y completamente las especificaciones, tanto en un equipo de escritorio domestico como en un grid-computer. También es interesante el recolector de basura que incorpora que se encarga de limpiar de la memoria las estructuras o variables que no se vayan a volver a usar durante la ejecución actual, evitando así el deterioro de la aplicación.

4.2. JDBC

Para conectar a Java con cualquier sistema gestor de bases de datos empleamos JDBC. JDBC es un marco de programación para los desarrolladores de Java necesitan tener un modelo de acceso a la información guardada en bases de datos, hojas de cálculo y archivos de texto planos. JDBC se utiliza comúnmente para conectar una aplicación de usuario con un sistema gestor de base de datos, sin importar qué gestor software de base de datos se utilice para almacenar la información. Empleando un dialecto del lenguaje de manipulación de datos SQL como medio de consulta o modificación de los mismos.

El API JDBC se presenta como una colección de interfaces Java y métodos de gestión de manejadores de conexión hacia cada modelo específico de base de datos. Un manejador de conexiones hacia un modelo de base de datos en particular es un conjunto de clases que implementan las interfaces Java y que utilizan los métodos de registro para declarar los tipos de localizadores a base de datos (URL) que pueden manejar.

Para utilizar una base de datos particular, el usuario ejecuta su programa junto con la biblioteca de conexión apropiada al modelo de su base de datos, y accede a ella estableciendo una conexión, para ello provee el localizador a la base de datos y los parámetros de conexión específicos. A partir de allí puede realizar con cualquier tipo de tareas con la base de datos a las que tenga permiso: consulta, actualización, creación, modificación y borrado de tablas, ejecución de procedimientos almacenados en la base de datos, etc.

4.3. LOG4J

Como herramienta de auditoría del funcionamiento de la aplicación, usamos Log4J. Log4j es una biblioteca de código abierto desarrollada en Java por la Apache Software Foundation y tiene como función el permitir a los desarrolladores elegir la salida y el nivel de prioridad de los mensajes a tiempo de ejecución y no a tiempo de compilación como es comúnmente realizado.

Es muy útil ya que nos permite durante la ejecución configurar la salida de errores desde a ficheros de diario de ejecución, como el envío a una determinada cuenta de correo electrónico, el almacenamiento en una base de datos y otros tantos destinos más.

Todo configurable sin tocar nada de código. Se eligió este sistema de auditoría frente a otros por ser un estándar de facto.

4.4. SQLITE

Las estadísticas que se obtienen y procesan durante todo el flujo de información, por defecto se almacenan en SQLite. SQLite es un sistema gestor de bases de datos relacional que cumple las especificaciones ACID[2]. A diferencia de los sistemas gestores de base de datos cliente-servidor, SQLite es un fichero de datos relacional que al que se accede mediante una biblioteca que ha de usarse desde el programa cliente de dicha información guardada en la base de datos, y no en un proceso externo a la aplicación.

[2] Un sistema de gestión de bases de datos cumple ACID quiere decir que el mismo cuenta con las funcionalidades necesarias para que sus transacciones tengan las características de atomicidad, consistencia, aislamiento, durabilidad.

Esto reduce la latencia en el acceso a la base de datos, debido a que las llamadas a funciones son más eficientes que la comunicación entre procesos. El conjunto de la base de datos (definiciones, tablas, índices, y los propios datos), son guardados como un sólo fichero estándar en la máquina host. Este diseño simple se logra bloqueando todo el fichero de base de datos al principio de cada transacción.

En este proyecto final de carrera hemos empleado SQLiteJDBC que implementa la versión 3 de SQLite y tiene una capacidad de hasta 2 terabytes de tamaño la inclusión de campos tipo BLOB. Obviamente no vamos a producir 2 terabytes de datos, es más si se va a producir un elevado valor de datos, por ejemplo de más de un centenar de megabytes es recomendable cambiar la configuración del Servicio para cambiar la línea que configura el driver de conexión para guardar las estadísticas, por otro sistema gestor de bases de datos más consistente.

4.5. CSV (Comma Separated Value)

El formato de intercambio de información elegido para que los datos recolectados puedan ser exportados hacia otra aplicación ha sido CSV.

El formato de fichero CSV (Comma Separated Value) de Valores Separados por Comas, se usa habitualmente para el intercambio de datos entre aplicaciones diferentes Los ficheros CSV son un tipo de ficheros estructurado en un formato sencillo para representar datos en forma de tabla, en las que los atributos de cada uno de los atributos de los registros se separan por comas y los propios registros se separan por saltos de línea. Este formato también es utilizado en Microsoft Excel, y se ha convertido en un formato estándar utilizado por muchas aplicaciones incluso en plataformas GNU/Linux.

El formato CSV tuvo una gran importancia y popularidad como formato de intercambio de datos antes de la aparición del estándar XML, de tal modo que llegó a convertirse en

un formato estándar de facto. Aunque en la actualidad está siendo desplazado por el estándar XML. Existen multitud de variantes del formato CSV, e incluso aplicaciones capaces de transformar ficheros de datos CSV en ficheros XML.

4.6. VISIONADO DE GRÁFICAS: JFREECHART VS GOOGLE CHART

Para la tarea de generación de los gráficos, se barajaron dos opciones. Google Chart y JFreeChart. Al final, se optaron por ambas soluciones. Google Chart se emplea en ChartServer que se ejecuta conjuntamente con la aplicación Servicio, si se encuentra implantada, y JFreeChart se emplea en la interfaz gráfica de la aplicación Analizador.

JFreeChart nos garantiza independencia del exterior. Google Chart nos garantiza mucha velocidad, pero tiene el inconveniente que sus invocaciones efectúan peticiones a URL de Google que es el que nos devuelve un gráfico en formato de imagen PNG. Google Chart, sorprendentemente y a pesar de que para su ejecución se necesita hacer una llamada a una URL remota, resulta más rápido que JFreeChart que es una biblioteca de nuestro proyecto.

Google Chart permite más tipos de gráfica, más fáciles de generar y más simples de invocar. Pero como ya he dicho, el invocar la URL remota, nos implica el enviar información a una entidad externa y la necesidad de conexión a internet para poder mostrar la gráfica, hecho que quizás no siempre nos sea posible o nos interese.

En resumidas cuentas, ambas opciones se han llegado a implementar, pero, por defecto la aplicación Analizador emplea JFreeChart, ChartServer en cambio es un complemento para la aplicación Servicio.

JFreeChart es una librería para gráficos desarrollada completamente en Java. JFreeChart facilita mostrar gráficos en nuestras aplicaciones, ya sean web o de escritorio. Entre las principales virtudes de esta biblioteca tenemos un API consistente y bien documentado

con soporte para un amplio rango de tipos de gráficas, también da soporte para varios tipos de salida de archivos de imagen como PNG y JPEG, y formatos gráficos vectoriales tales como PDF, EPS y SVG. JFreeChart es una biblioteca que está distribuida bajo la licencia LGPL, luego gratuita y que permite el uso en aplicaciones propietarias.

Google Chart es uno de los últimos *juguetes* de Google. Su nombre completo es Google Chart Imagen Generator, una herramienta por la cual se nos permite crear, vía petición http:// a la API, unas gráficas que podrán ser lineales, de barras, de tarta, de Venn y de dispersión, y que podemos insertar en nuestra web. En el caso de este proyecto, ChartServer genera a partir de los datos de estadísticas las distintas URL que necesita para mostrar los datos, efectúa las peticiones, maqueta los resultados y los publica en modo de página HTML, para que el usuario de ChartServer se conecte a este por medio de un navegador web y visualice los resultados. Volviendo a Google Chart, para obtener las gráficas tan sólo deberemos de crear una URL pasando una serie de parámetros en la misma y se nos devolverá en forma de gráfica en formato PNG. Tan solo tiene como restricción que si vas a emplear más de 250000 peticiones diarias, se lo comunique para que no te tomen como una maniobra de ataque. Para más información se puede consultar http://code.google.com/intl/es/apis/chart/ .

4.7. Entorno gráfico: Swing + JDesktop

El entorno gráfico Swing de la aplicación Analizado está construido con JDesktop. JDesktop es un framework al estilo de los framework web tipo Struts o Spring. Las interfaces gráficas son clases Java que se apoyan en un fichero XML donde se guardan las características visuales que se mostraran y otro fichero de propiedades que guardará metadatos de los elementos visuales. Como los framework citados anteriormente se basan en el paradigma modelo-vista-controlador.

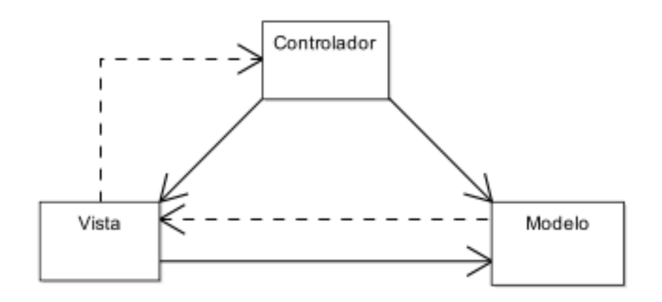


Ilustración 10 – http://es.wikipedia.org/wiki/Modelo_Vista_Controlador

Modelo: Es la representación específica de la información con la cual el sistema opera. Esta función la hace la clase Java cargando las características del fichero de propiedades a petición.

Vista: Presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario. Se genera a partir del XML, que es procesado por la clase Java.

Controlador: Responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista. La clase Java se encarga de esta función, siendo esta la que responde a los eventos. Las bibliotecas del framework simplifican estas operaciones.

Por la función vital que ocupa la clase Java, tiene más parecido de entre todos los framework web con los que he trabajado, con Spring.

4.8. IDE: NETBEANS

Y una pequeña nota ahora al orquestador de todo el desarrollo, NetBeans. NetBeans simplemente se puede definir como la mejor IDE, entorno de desarrollo integrado, de la actualidad. NetBeans es multiplataforma y corre bajo Java. NetBeans es un proyecto de código abierto de gran éxito con una gran base de usuarios. Sun MicroSystems fundó el proyecto de código abierto NetBeans en junio 2000 y continúa siendo el patrocinador

principal de los proyectos. NetBeans comenzó como un proyecto estudiantil en Republica Checa (originalmente llamado Xelfi), en 1996 bajo la tutoría de la Facultad de Matemáticas y Física en la Universidad de Charles en Praga. La meta era escribir un entorno de desarrollo integrado para Java parecida a la de Delphi. Xelfi fue el primer entorno de desarrollo integrado escrito en Java. Con soporte para Java SE, aplicaciones Web bajo Java EE, Java ME para móviles, Java FX, desarrollo SOA con OpenESB, Ruby, C/C++, PHP, Groovy, Javascript, Python, y la integración con GlassFish, OpenESB y Apache Tomcat, NetBeans se puede considerar el entorno de desarrollo integrado más completo existente para Java, además de otros cuantos lenguajes más.

5. DISEÑO DE LA SOLUCIÓN

A lo largo de este capítulo pasaremos a describir de forma más detallada cada una de los componentes que forman este proyecto final de carrera. Realizando un amplio análisis de su diseño y de sus funciones.

5.1. ENUMERACIÓN DE COMPONENTES

Para empezar, hemos de enunciar los distintos componentes que forman parte del proyecto, así como una breve introducción a ellos.

El proyecto consta de cinco componentes. Estos son tres aplicaciones, una biblioteca de funciones de red y un complemento para una de las aplicaciones. Sus nombres son Servicio, Analizador, Cliente de Estadisticas, ChartServer y TCP.

5.1.1. SERVICIO

Vamos a empezar a hablar del componente Servicio. Servicio es una pequeña aplicación que se lanza como servicio de sistema y que efectúa un conjunto de sentencias SQL o PL/SQL sobre un sistema gestor de bases de datos.

Servicio, mediante un sistema de planificación de tareas, de formato similar al comando cron de Linux, carga tareas según una planificación dada y son lanzados cuando les corresponde. Para cada tarea lanzada, anota el instante de la petición y de la respuesta, calcula el tiempo que se ha tardado en efectuar dicha petición y los almacena en una

tabla, llamada Tabla de Estadísticas. Normalmente dicha tabla se almacena en un fichero SQLite, aunque puede ser configurado para que se guarde en cualquier sistema gestor de bases de datos o cualquier sistema de almacenamiento accesible con JDBC.

Servicio además almacena diarios de sus ejecuciones gracias a Log4J, siendo este configurable para el envío de avisos por correo electrónico si fuera necesario.

Decir también que en los primeros prototipos empleaba iBATIS es un framework basado en capas desarrollado por Apache Software Foundation, que se ocupa de la capa de Persistencia y se sitúa entre la lógica de Negocio y la capa de la Base de Datos, pero siguiendo desarrollos propios, la versión actual emplea clases propias de este proyecto final de carrera para estos menesteres ya que con la clases integradas en el propio proyecto hemos conseguido mejores rendimientos que con el framework anteriormente mencionado.

5.1.2. ChartServer

Como complemento a Servicio, se le puede incluir en su despliegue el que llamamos ChartServer.

ChartServer es un servidor HTTP que publica unas gráficas de estadísticas básicas, en tiempo de ejecución. Cuando un usuario, se conecta al puerto HTTP de ChartServer, el hilo de ejecución correspondiente a esa sesión recolecta datos del fichero de estadísticas actual, genera una URL que es enviada a la API de Google Chart, esta nos devuelve un gráfico PNG por cada una de las URL solicitadas, el hilo de ejecución genera un fichero HTML con la inclusión de los correspondientes gráficos recibido y los envía en el GET al navegador del usuario, y todo esto en un tiempo ínfimo.

5.1.3. *JOSEJAMILENA.PFC.SERVIDOR.TCP*

Pasamos ahora a comentar la biblioteca josejamilena.pfc.servidor.tcp también llamada TCP. Esta biblioteca contiene funciones tales como:

- Suma de verificación generando el polinomio de comprobación de redundancia cíclica de 32 bits (CRC32) y se suele utilizar para validad la integridad de los datos transmitidos.

- Comunicación por TPC/IP, de ahí su nombre.

- Utilidades de copia rápida de ficheros mediante channles. Los channels representan conexiones a entidades que son capaces de realizar operaciones I/O, tales como ficheros, sockets,... ofreciendo unos rendimientos muy elevados en la comunicación.

5.1.4. *CLIENTE ESTADÍSTICAS*

Esta aplicación emplea la librería TCP, para conectarse al componente Servicio, y obtiene una copia local de los datos publicados. Funciona desde línea de comandos.

5.1.5. *ANALIZADOR*

El Analizador es la interfaz gráfica de análisis de los datos almacenados en el fichero de estadísticas que es generado por Servicio.

Analizador abre[3] un fichero de estadísticas, o las recibe por red. Una vez obtenido el origen de los datos, Analizador se conecta a ellos mediante JDBC. En este paso, hay que indicar que el formato nativo de los datos que puede abrir el componente Analizador es SQLite versión 3 generado por SQLiteJDBC. Los ficheros que puede abrir son ficheros relacionales en este formato. Si los datos que genera Servicio se han guardado en un sistema de bases de datos relacional como puede ser Oracle, para obtener los datos a analizar, hemos de hacerlo haciendo la llamada a Servicio para que el nos envíe los datos y Analizador los almacene en el formato nativo de este, es decir, SQLite. Una vez en este formato se puede trabajar con ello, exportarlos como .CSV,…

Una vez que se ha conectado a los datos al Analizador, ya se puede tratar la información.

Analizador puede generar gráficas por Sistema de Gestor de Bases de Datos, por Script y por Host Cliente. En dichas gráficas se muestran la evolución de la latencia de las peticiones realizadas por Servicio al sistema gestor de bases de datos a analizar a lo largo del tiempo.

Analizador tiene también la capacidad de exportar los datos a formato .CSV, lo que facilita el poder analizarlos en Microsoft Excel o en SPSS.

5.2. Diseño de Servicio

Vamos ahora a mostrar los distintos diagramas del diseño del componente Servicio.

[3] Cuando nos referimos a abrir, es si las estadísticas vienen almacenadas en un fichero SQLite. Si los datos se han guardado en un sistema de bases de datos relacional como puede ser Oracle, para obtener los datos a analizar, hemos de hacerlo haciendo la llamada a Servicio para que el nos envíe los datos y Analizador los almacene en el formato nativo de este, es decir, SQLite.

5.2.1. DIAGRAMA DE CLASES

Vamos a ver un diagrama principal, y luego lo desglosaremos.

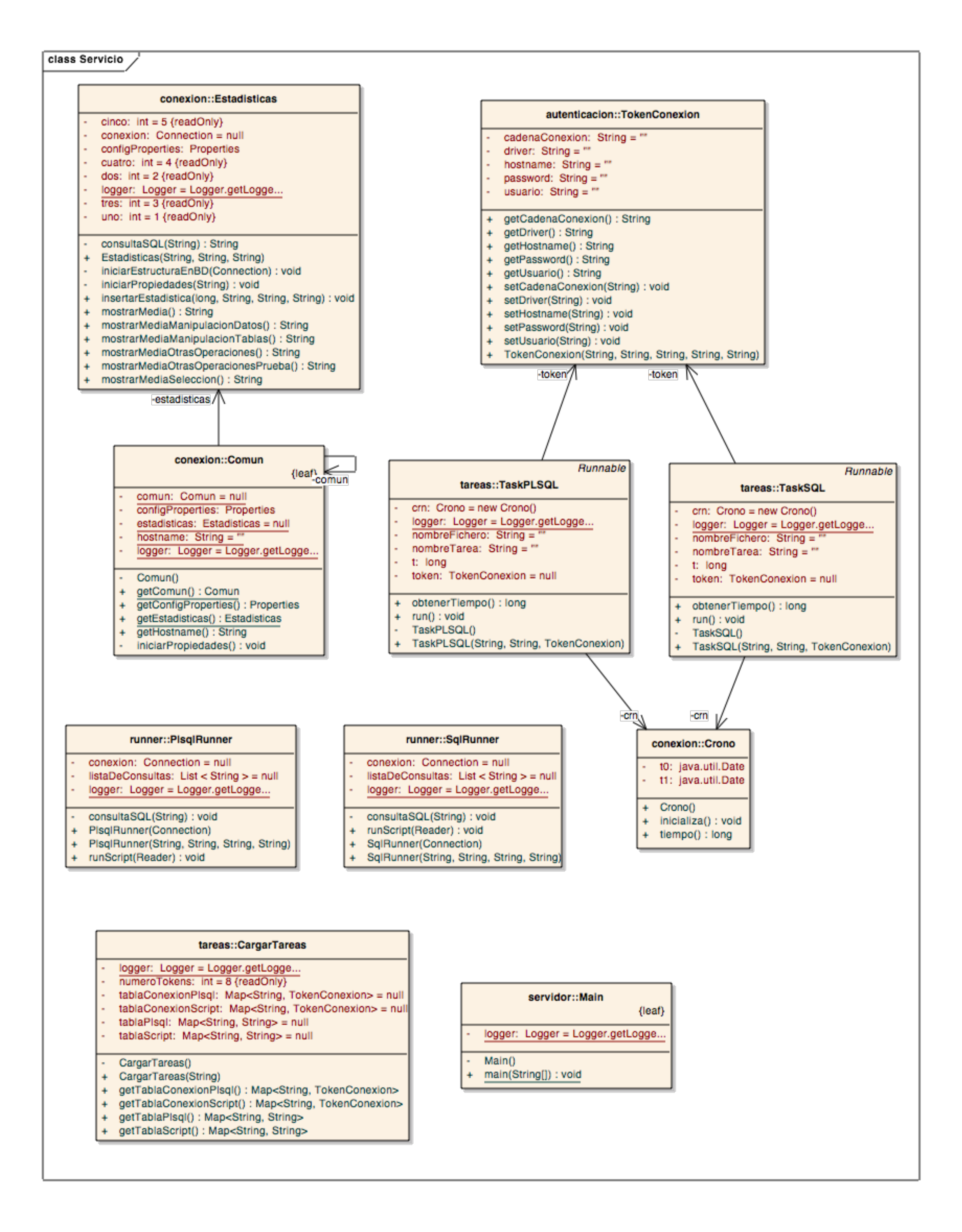


Ilustración 11 – Diagrama de clases de Servicio

5.2.1.1. Paquete josejamilena::pfc:servidor

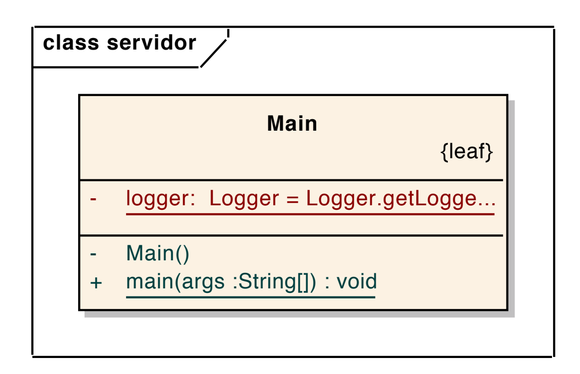


Ilustración 12 – Diagrama de clases de jose::pfc::servidor

5.2.1.2. Paquete josejamilena::pfc:servidor::conexión

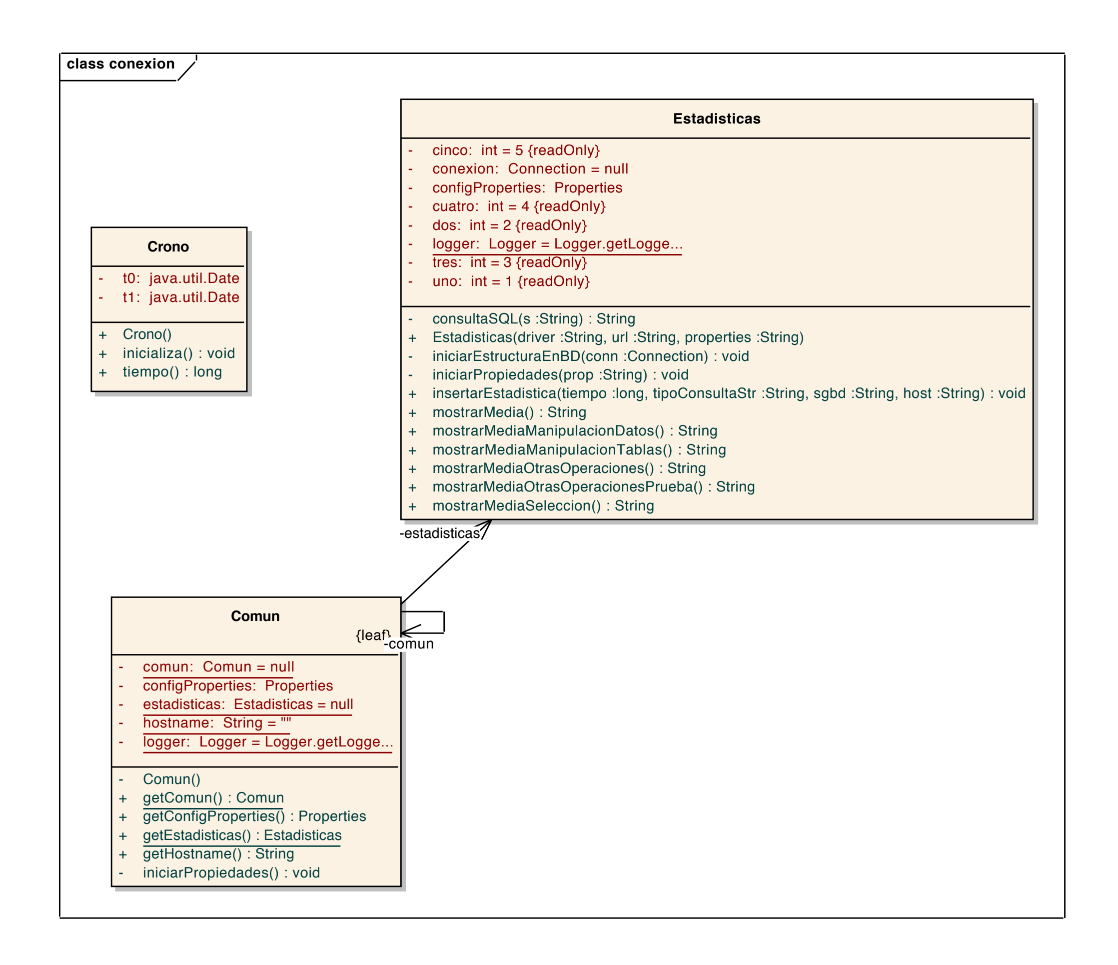


Ilustración 13 – Diagrama de clases de josejamilena::pfc::servidor::conexion

5.2.1.3. Paquete josejamilena::pfc:servidor::tareas

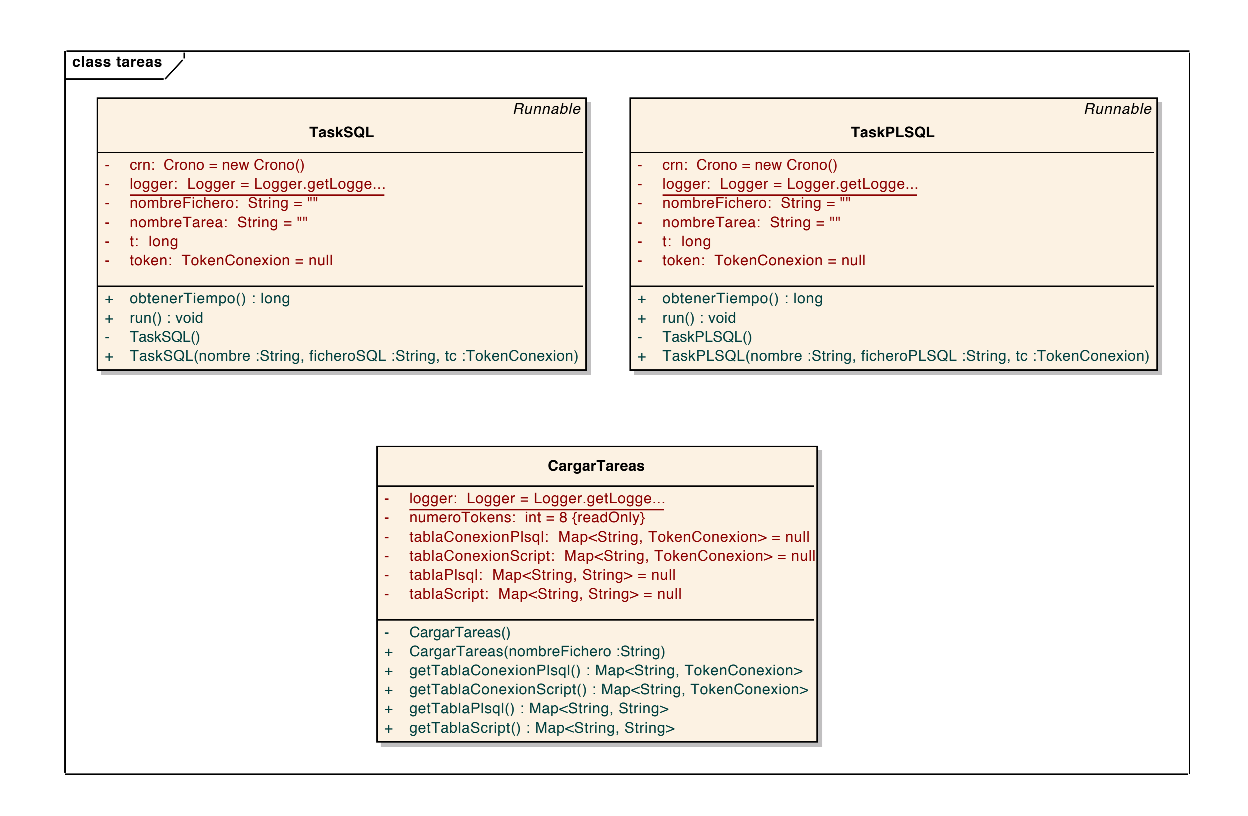


Ilustración 14 – Diagrama de clases de josejamilena::pfc::servicio::tareas

5.2.1.4. Paquete josejamilena::pfc:servidor::tareas::autenticacion

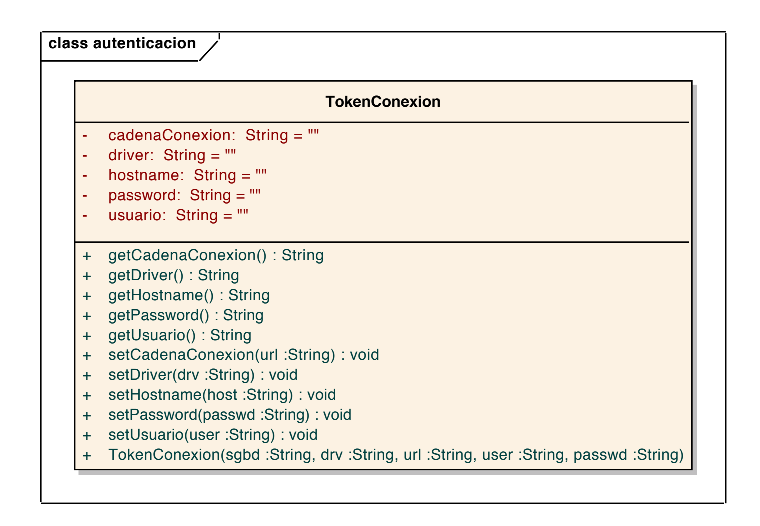


Ilustración 15 – Diagrama de clases de josejamilena::pfc::servidor::tareas::autenticacion

5.2.1.5. Paquete josejamilena::pfc:servidor::tareas::runner

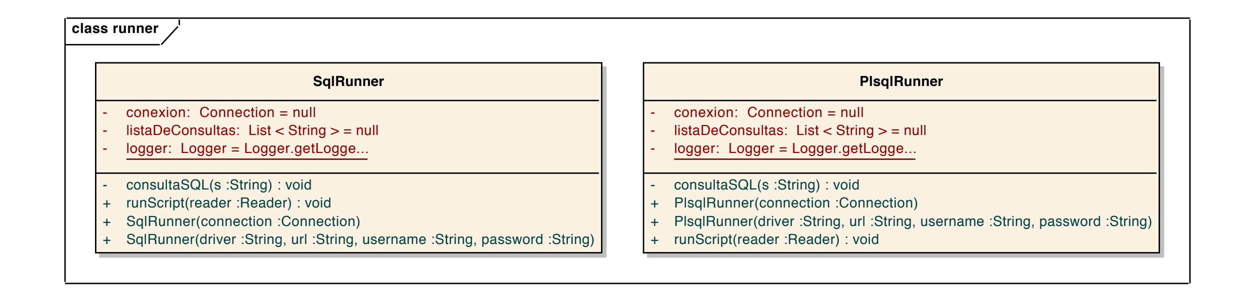


Ilustración 16 – Diagrama josejamilena::pfc::servicio::tareas::runner

5.2.2. DIAGRAMAS DE CASOS DE USO

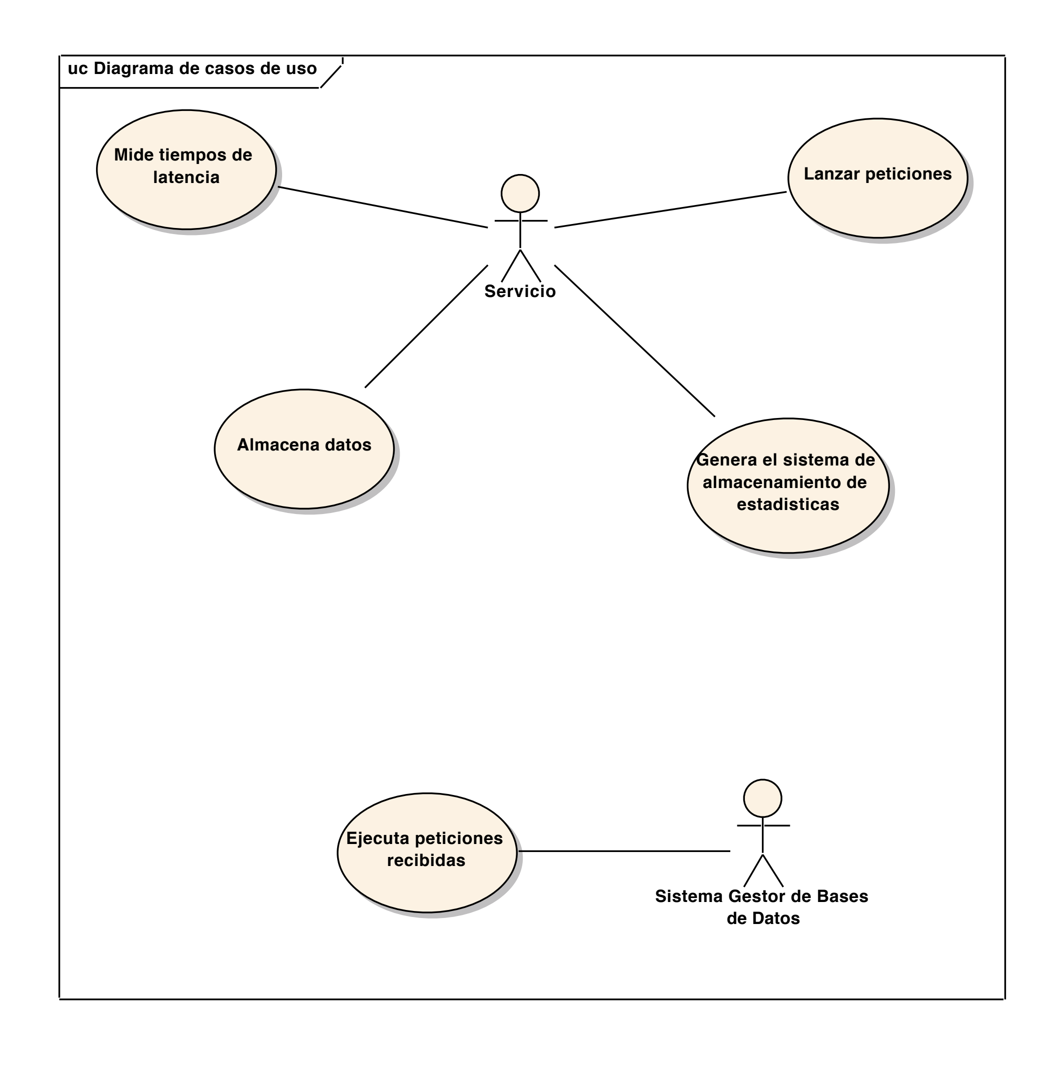


Ilustración 17 – Diagrama de casos de use de Servicio

5.2.3. DIAGRAMAS DE COMPONENTES

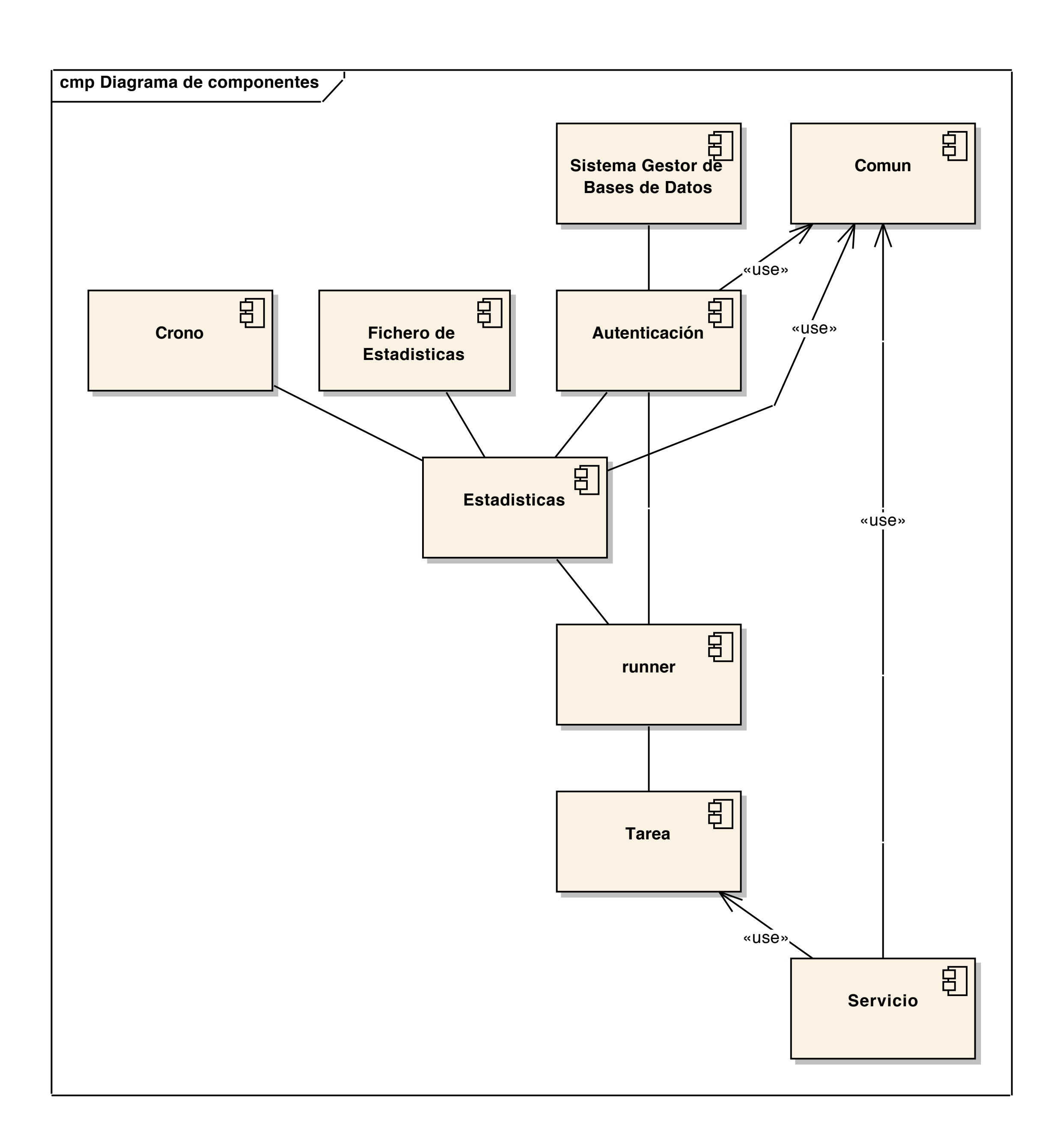


Ilustración 18 – Diagrama de componentes de Servicio

5.2.4. DIAGRAMAS DE COMUNICACIONES

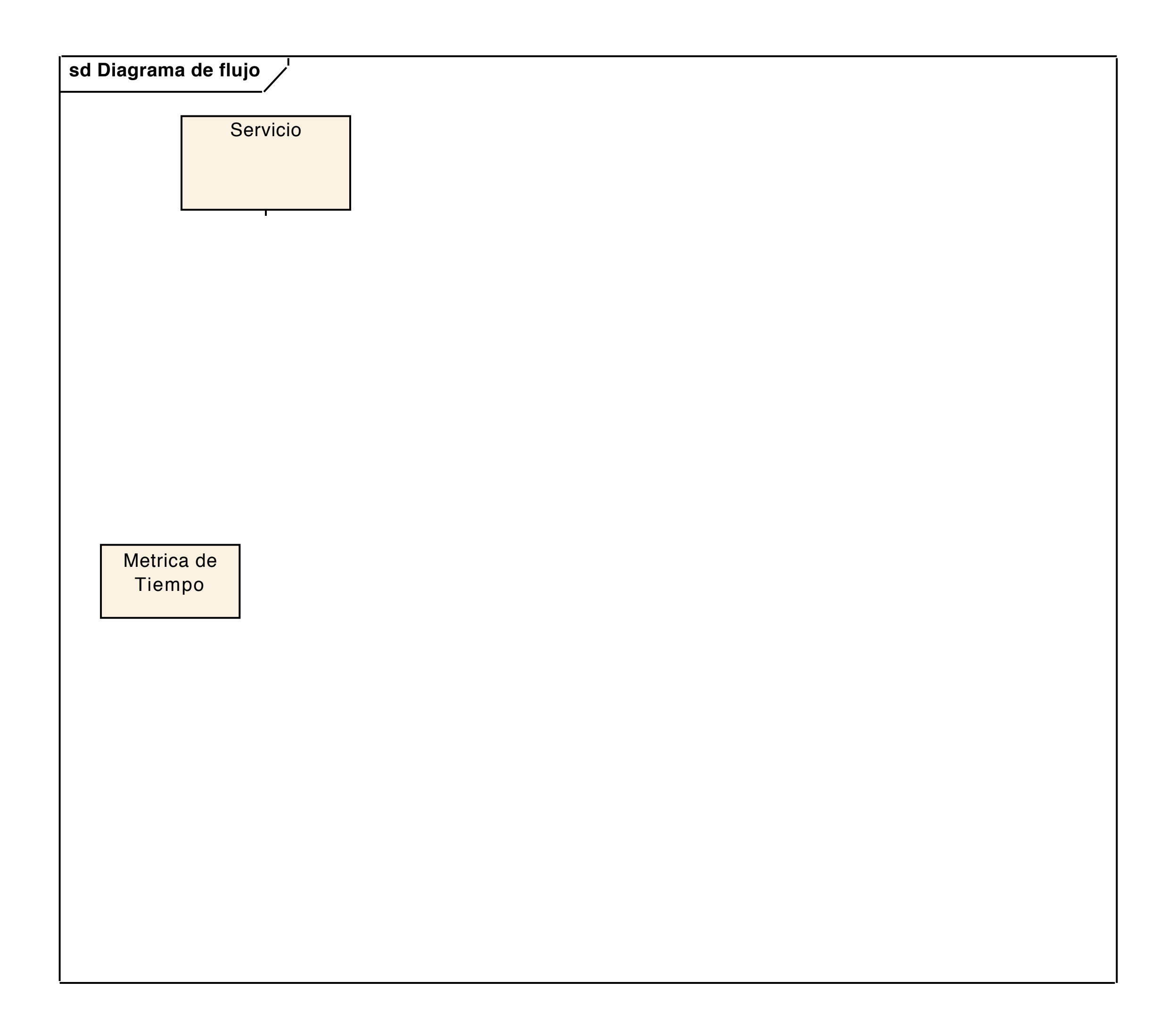


Ilustración 19 – Diagrama de comunicaciones

5.2.5. Diagramas de actividades

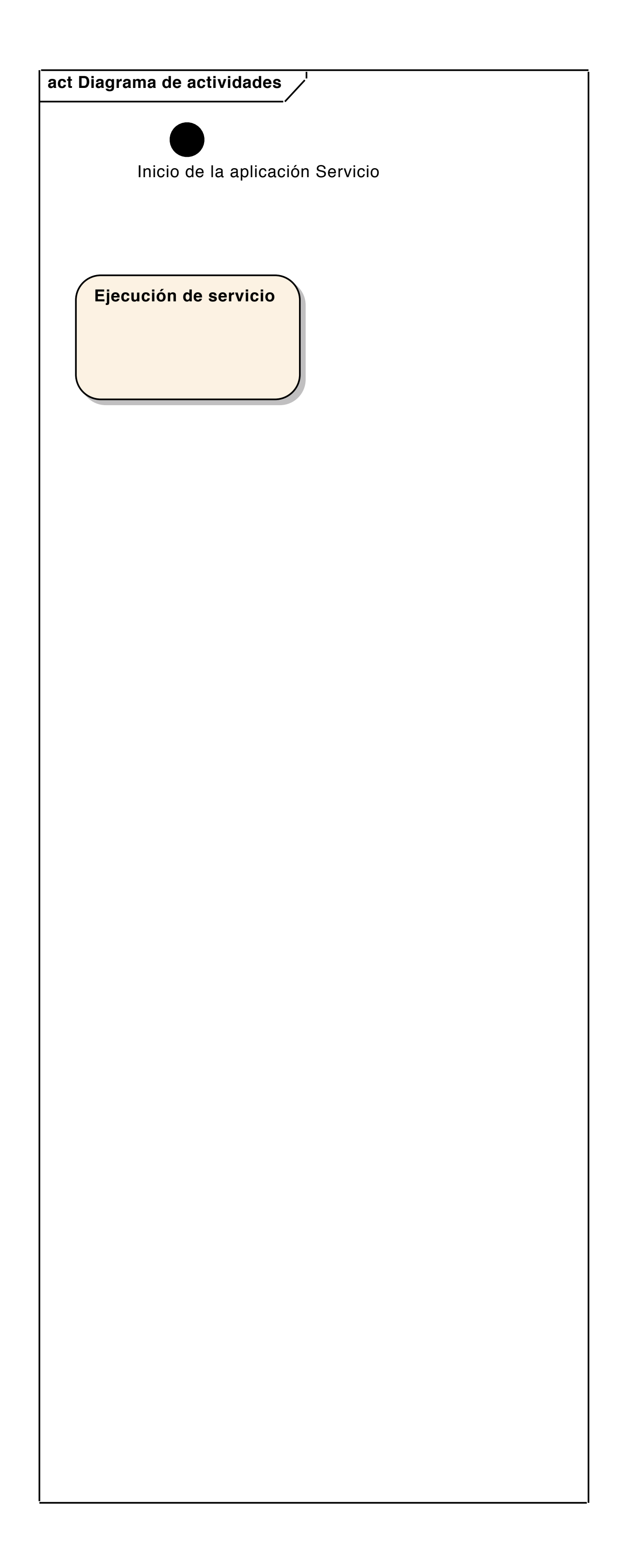


Ilustración 20 – Diagrama de actividades

5.3. DISEÑO DE CHARTSERVER

5.3.1. DIAGRAMAS DE CLASES

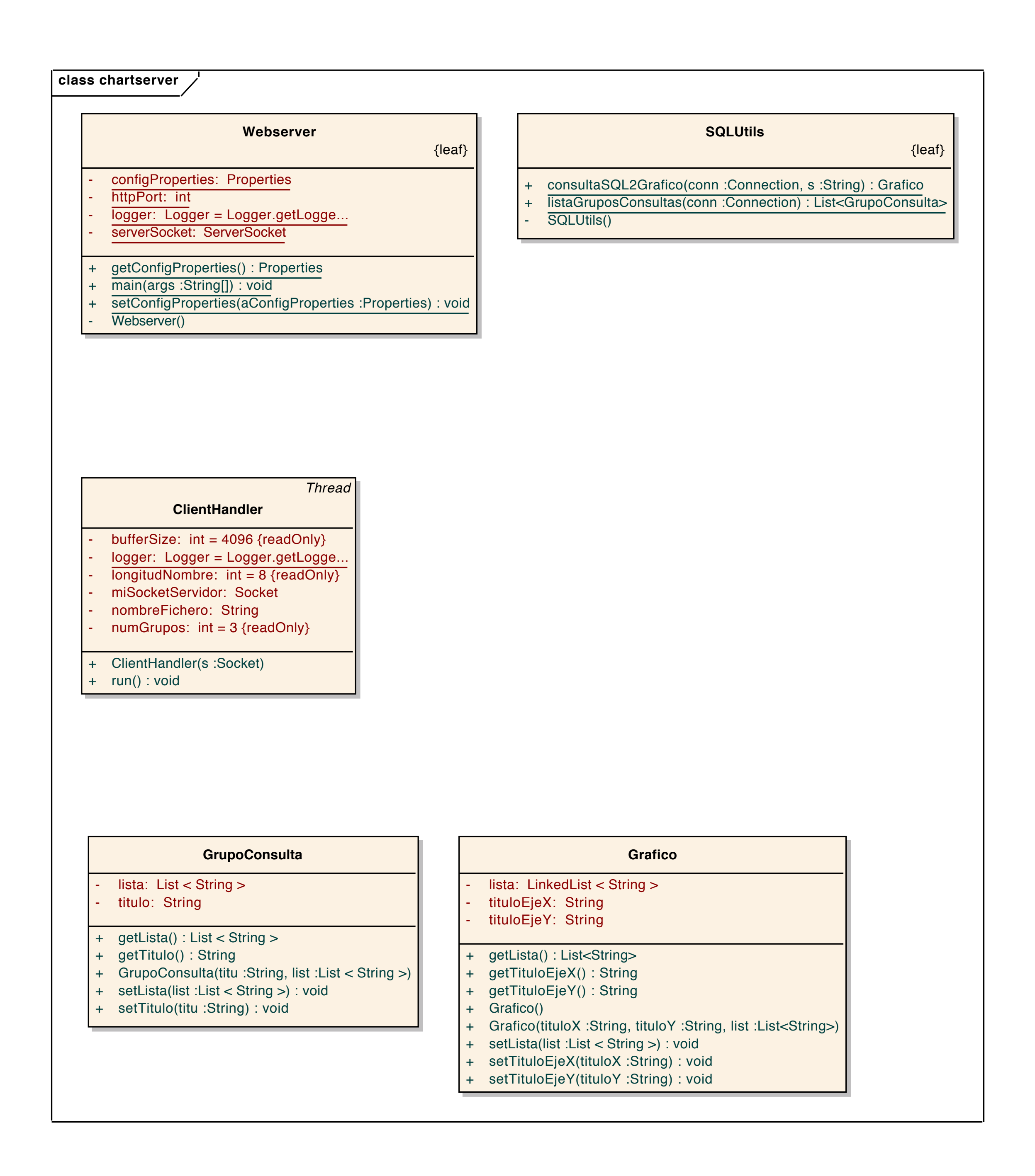


Ilustración 21 – Diagrama de clases de ChartServer

5.3.2. DIAGRAMAS DE COMPONENTES

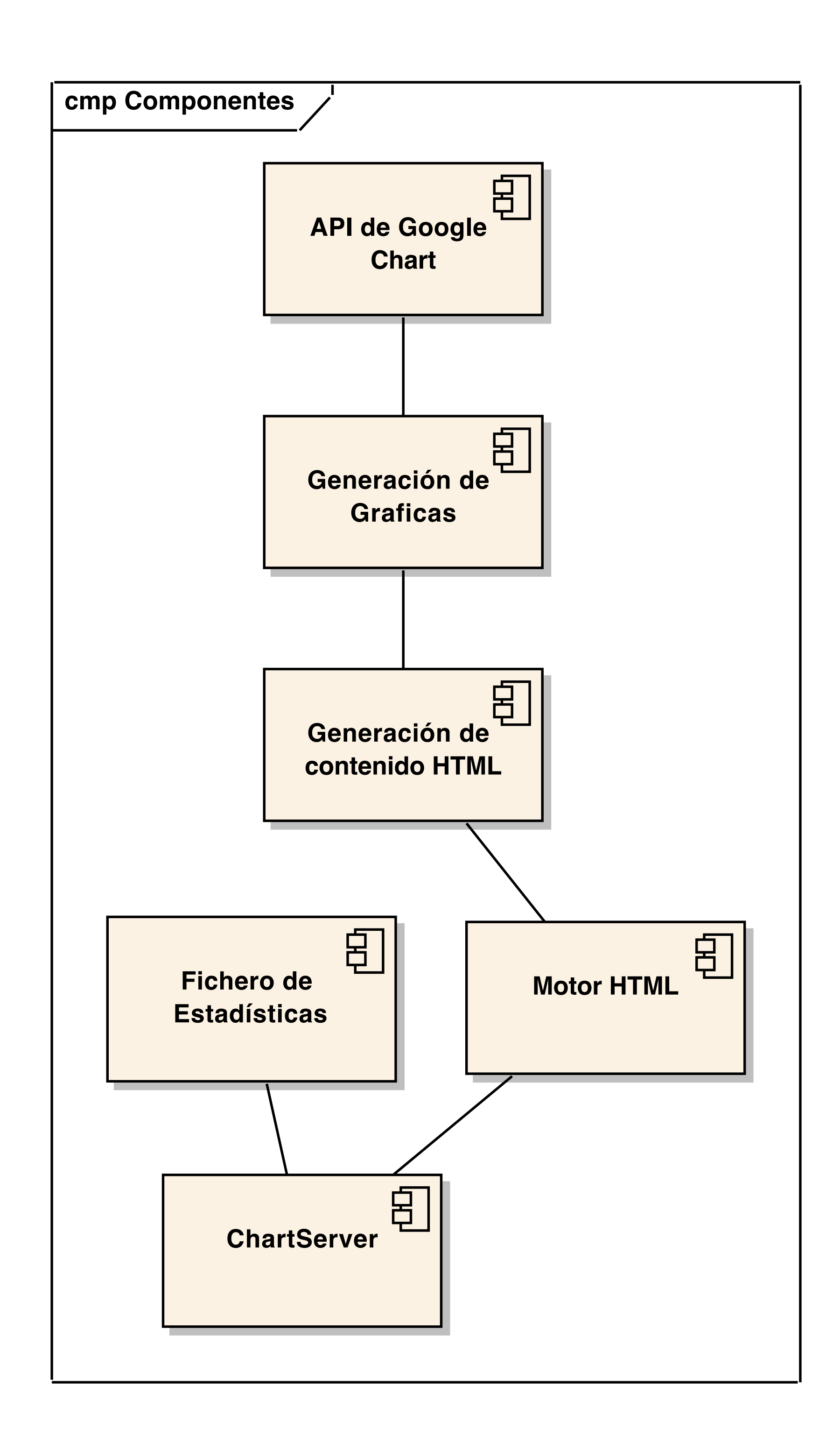


Ilustración 22 – Diagrama de componentes de ChartServer

5.3.3. DIAGRAMAS DE COMUNICACIONES

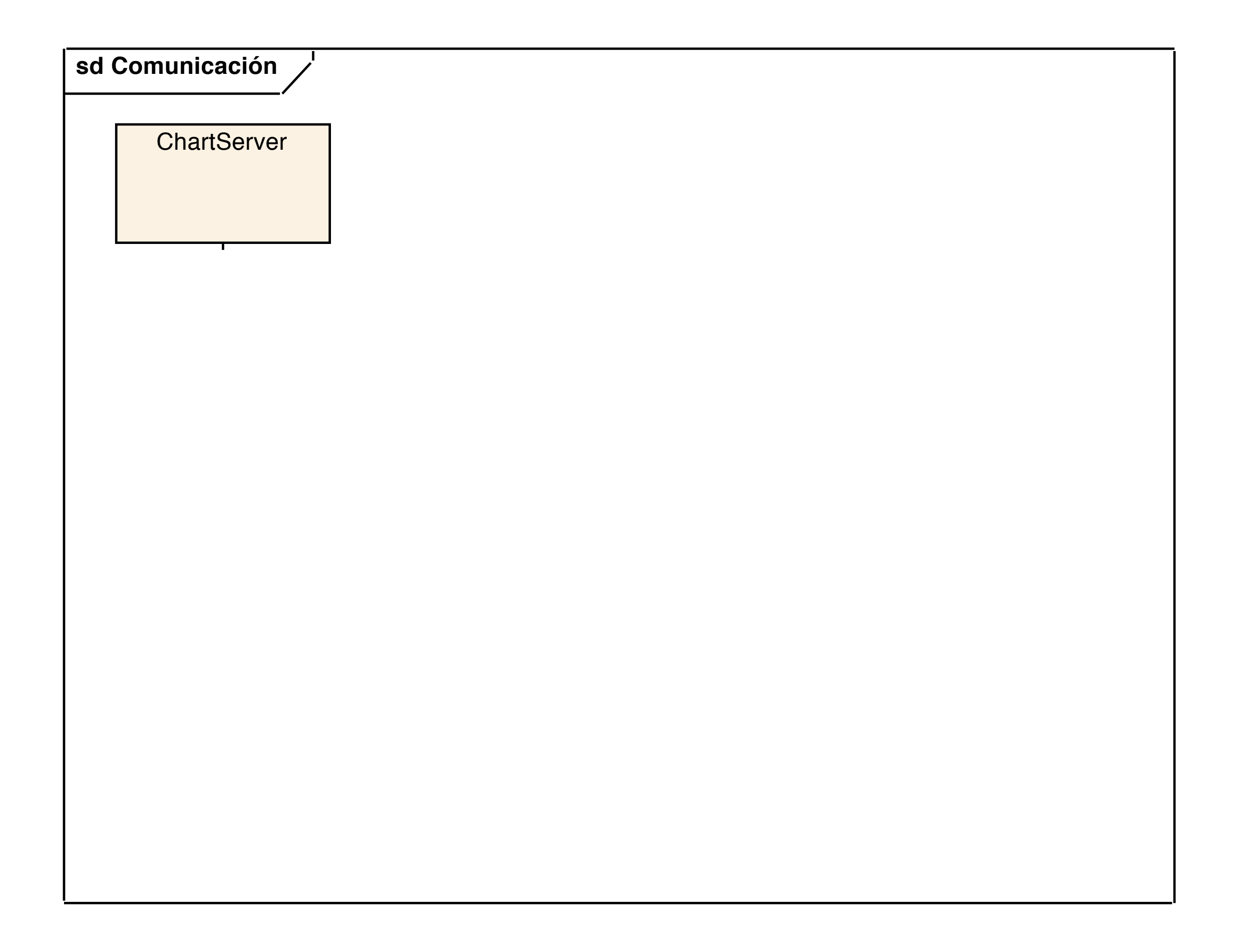


Ilustración 23 – Diagrama de comunicaciones de ChartServer

5.3.4. DIAGRAMAS DE ACTIVIDAD

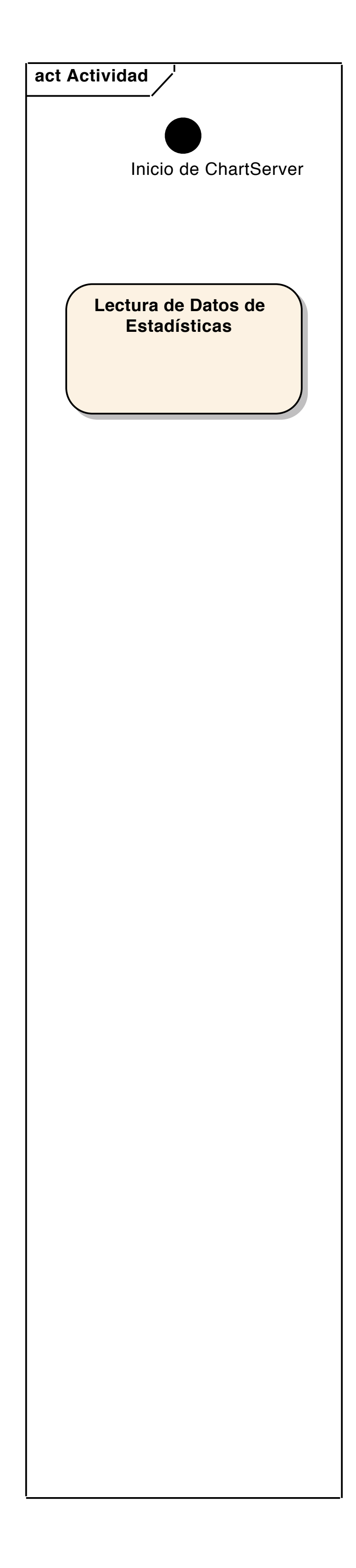


Ilustración 24 – Diagrama de actividades de ChartServer

5.4. Diseño de la biblioteca de funciones TCP

5.4.1. DIAGRAMAS DE CLASES

5.4.1.1. josejamilena::pfc::servidor::crypto::easy::checksum

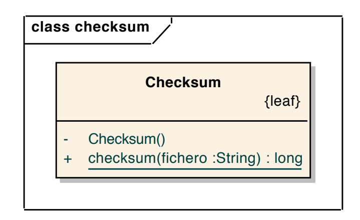


Ilustración 25 – Diagrama de clases de josejamilena::pfc::servidor::crypto::easy::checksum

5.4.1.2. josejamilena::pfc::servidor::tcp

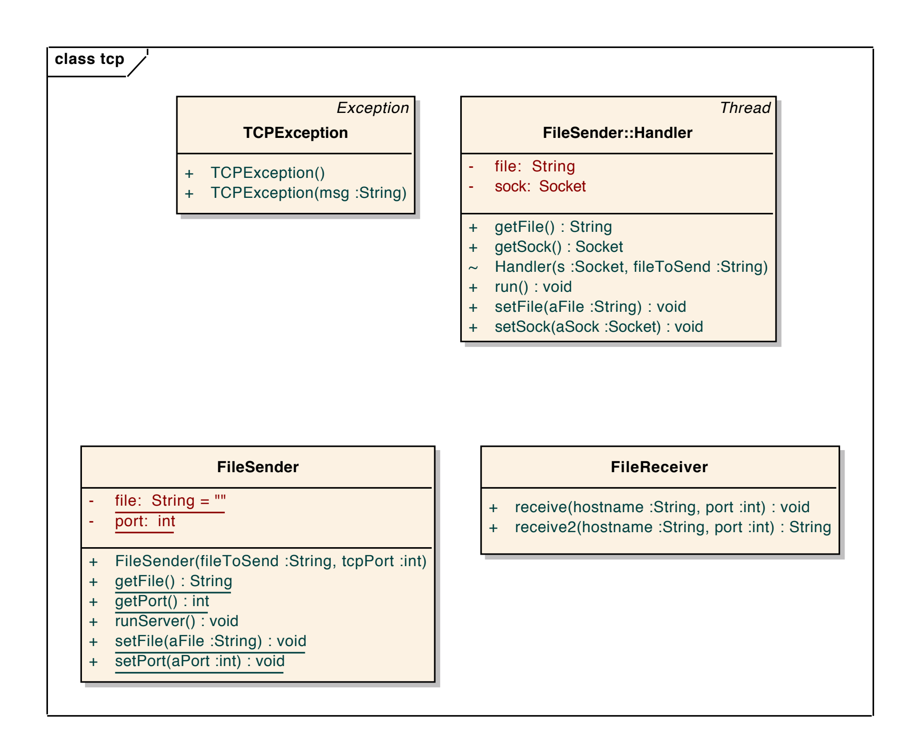


Ilustración 26 – Diagrama de clases de josejamilena::pfc::servidor::tcp

5.4.1.3. josejamilena::pfc::servidor::util

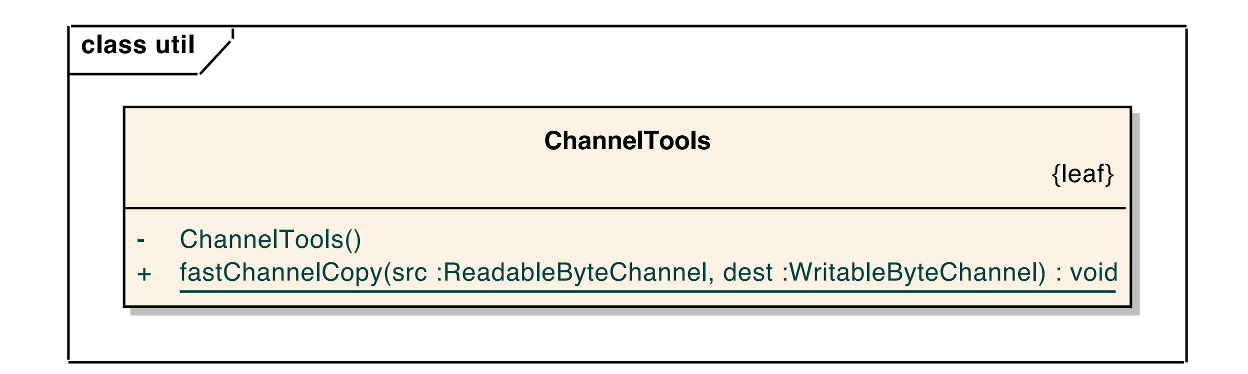


Ilustración 27 – Diagrama de clases de josejamilena::pfc::servidor::util

5.5. DISEÑO DEL CLIENTE DE ESTADÍSTICAS

5.5.1. DIAGRAMAS DE CLASES

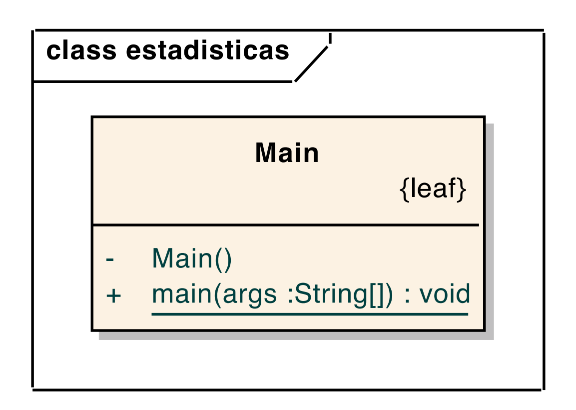


Ilustración 28 – Diagrama de clases de Cliente de Estadisticas

5.5.2. DIAGRAMAS DE COMUNICACIONES

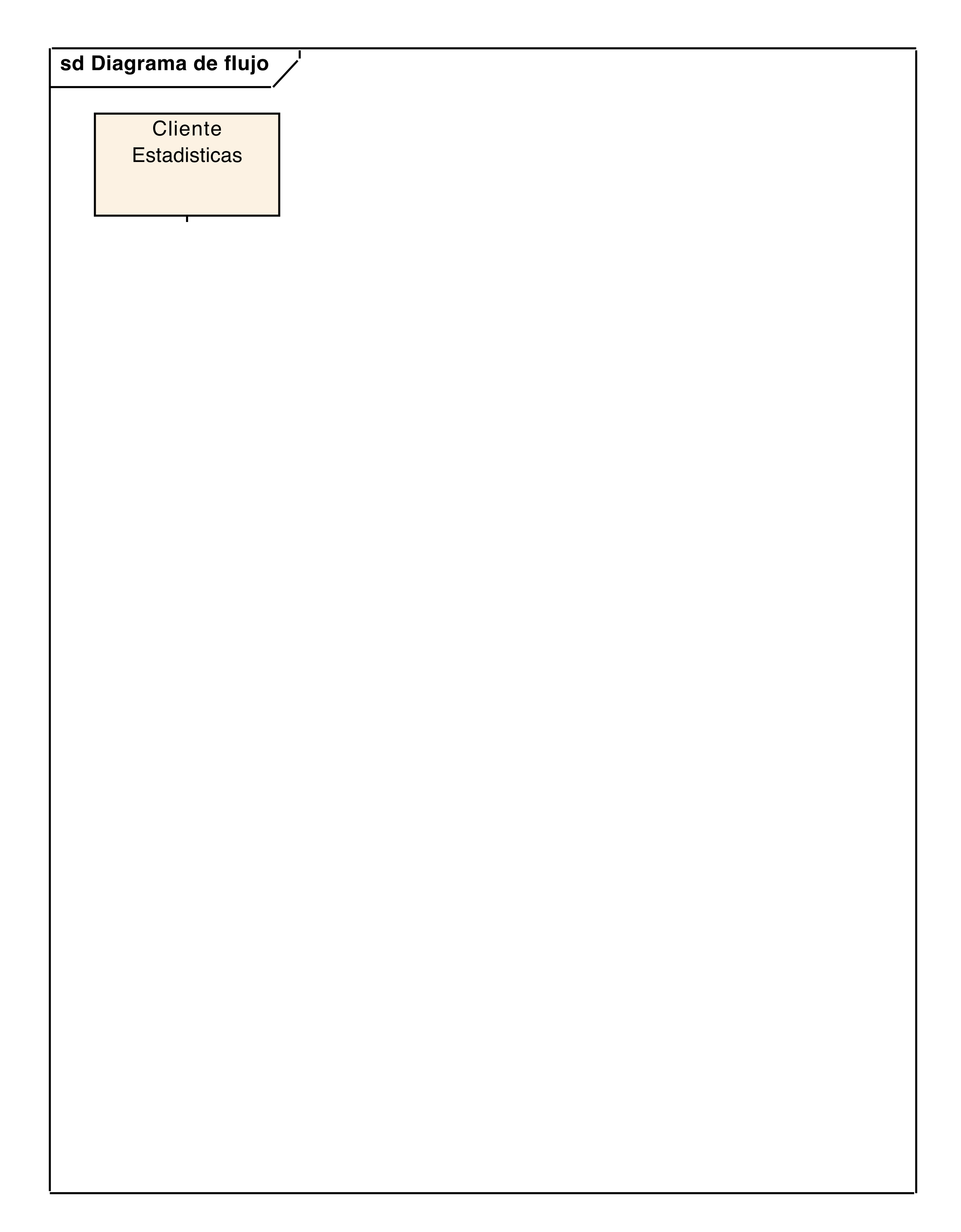


Ilustración 29 – Diagrama de comunicaciones de Cliente de Estadisticas

5.6. DISEÑO DE ANALIZADOR

5.6.1. DIAGRAMAS DE CLASES

5.6.1.1. josejamilena::pfc::analizador

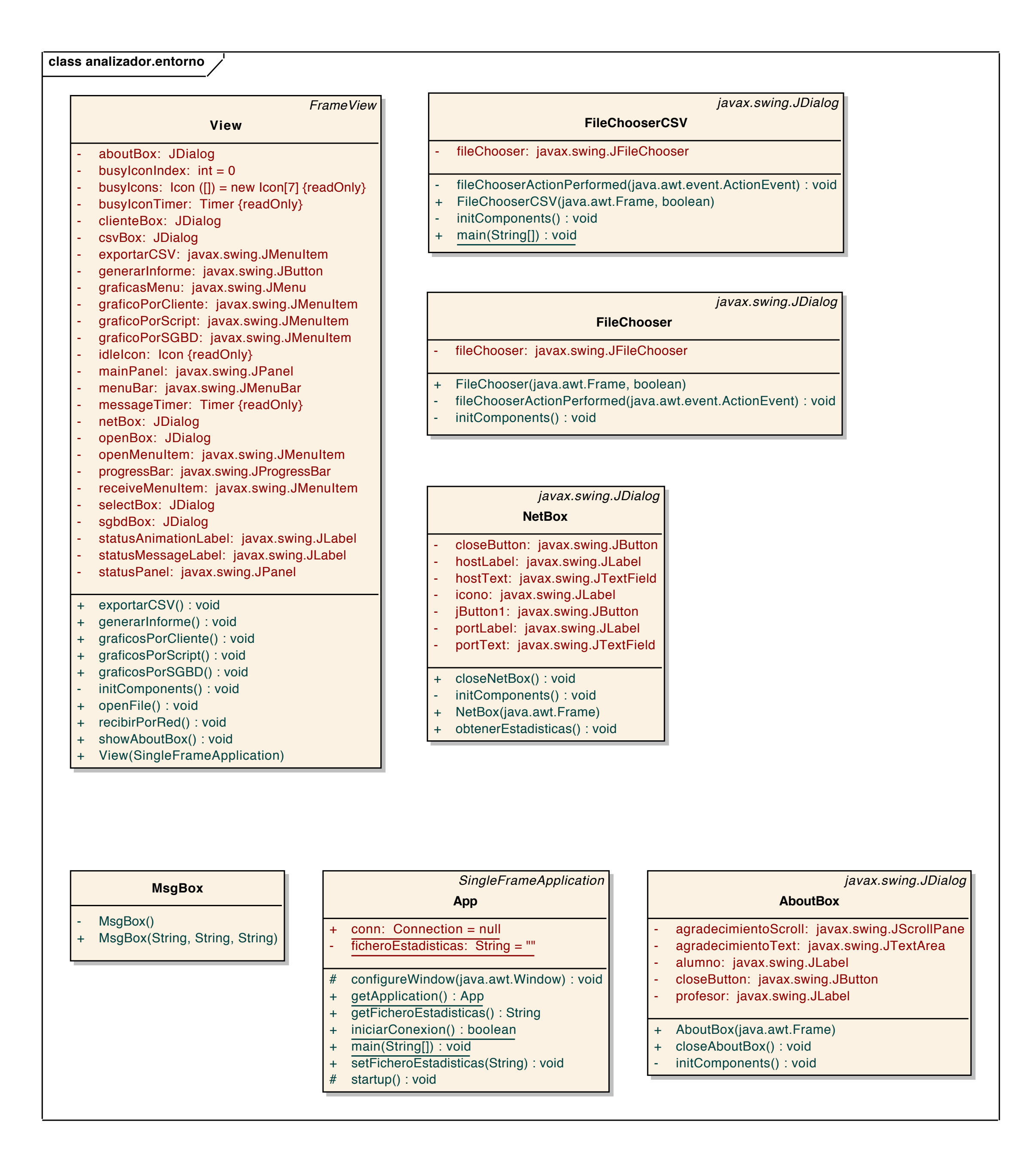


Ilustración 30 – Diagrama de clases de Analizador (parte 1)

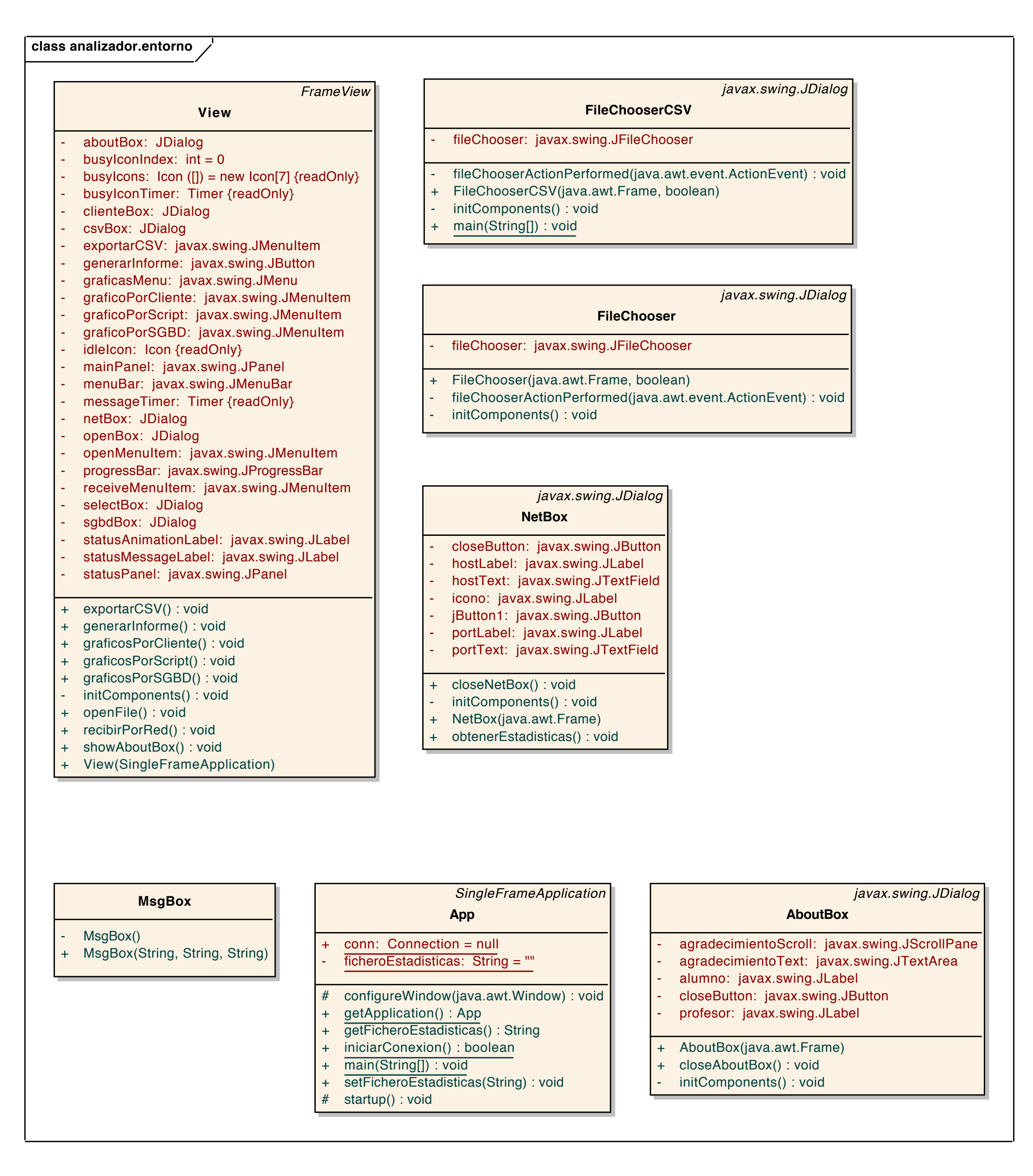


Ilustración 31 – Diagrama de clases de Analizador (parte 2)

5.6.2. JOSEJAMILENA::PFC::ANALIZADOR::SQL

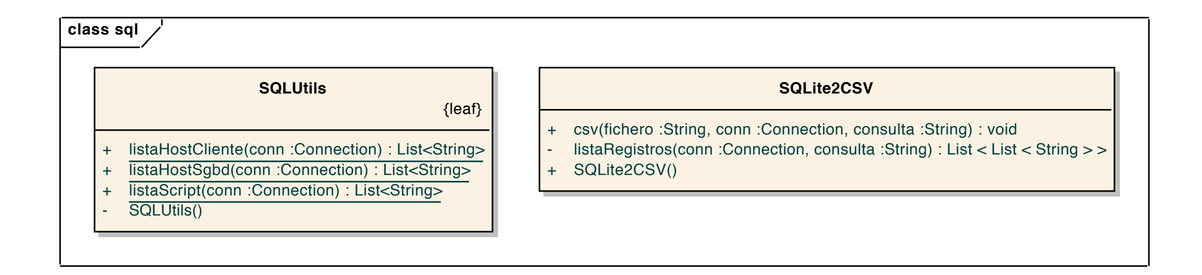


Ilustración 32 – Diagrama de clases de josejamilena::pfc::analizador::sql

5.6.3. DIAGRAMAS DE CASOS DE USO

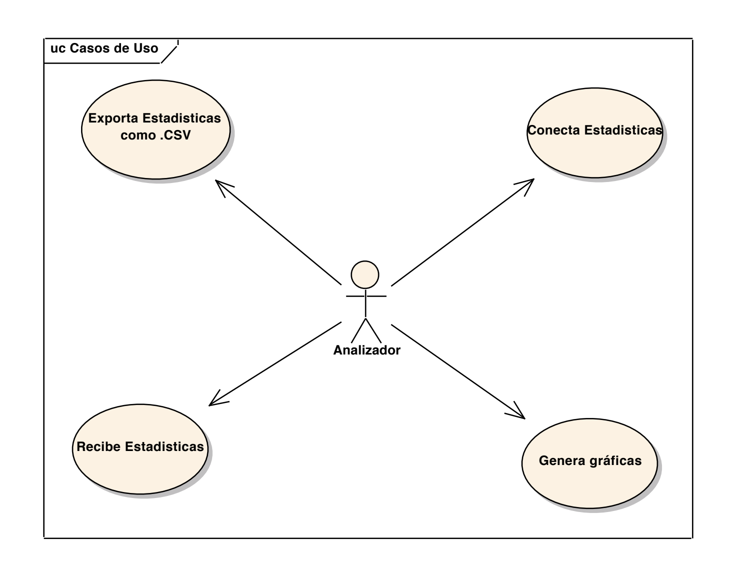


Ilustración 33 – Diagrama de casos de uso de Analizador

5.6.4. DIAGRAMAS DE COMUNICACIONES

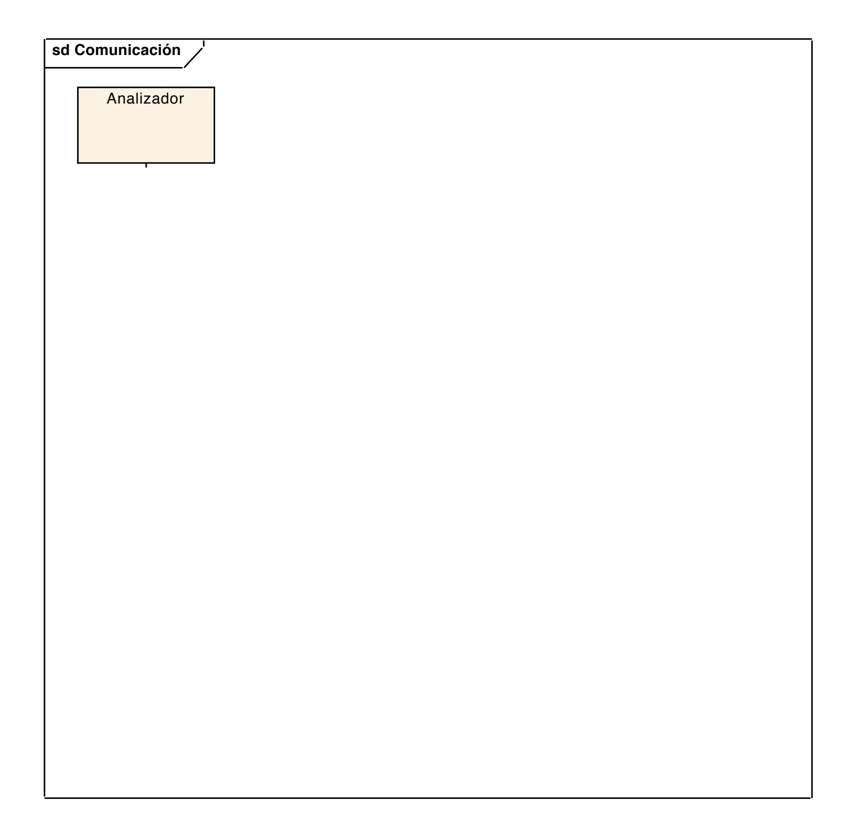


Ilustración 34 – Diagrama de comunicaciones de Analizador

5.6.5. DIAGRAMAS DE ACTIVIDADES

5.6.5.1. Generar una gráfica por SGBD

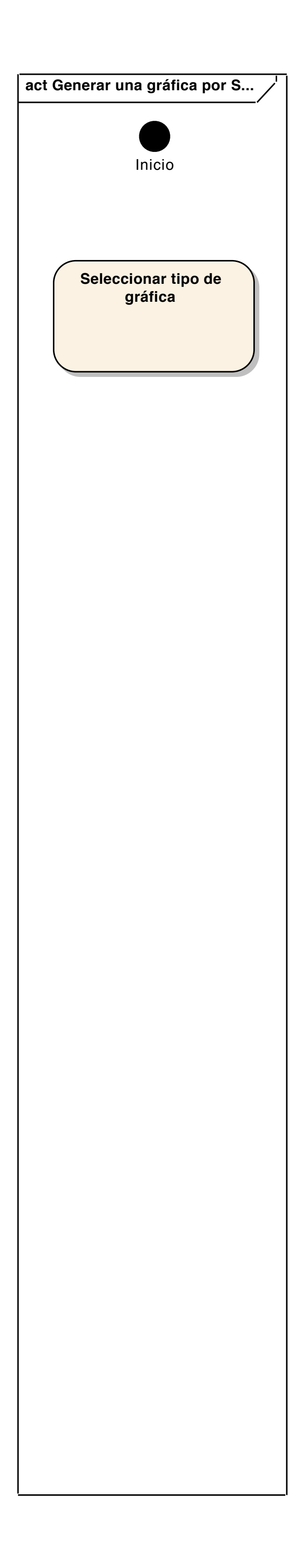


Ilustración 35 – Diagrama de Actividades de Analizador (Generar gráfica por SGBD)

5.6.5.2. Generar una gráfica por Script

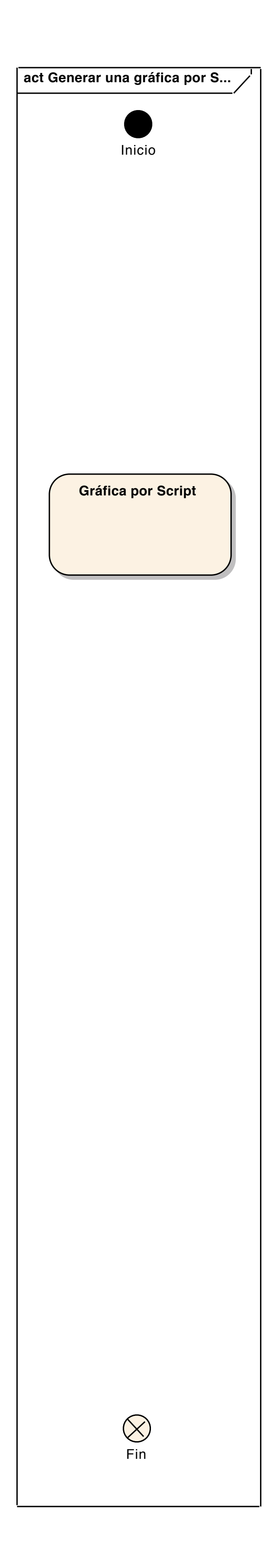


Ilustración 36 – Diagrama de Actividades de Analizador (Generar gráfica por Script)

5.6.5.3. Generar una gráfica por Host Cliente

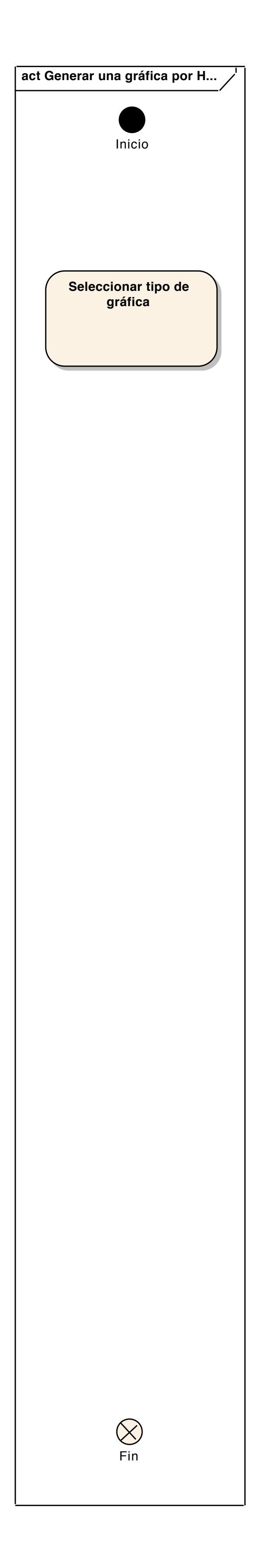


Ilustración 37 – Diagrama de Actividades de Analizador (Generar gráfica por Host Cliente)

5.6.5.4. Exportar datos a archivo .CSV



Ilustración 38 – Diagrama de Actividades de Analizador (Exportar como fichero de valores separados por coma)

5.6.5.5. Abrir archivo de Estadísticas

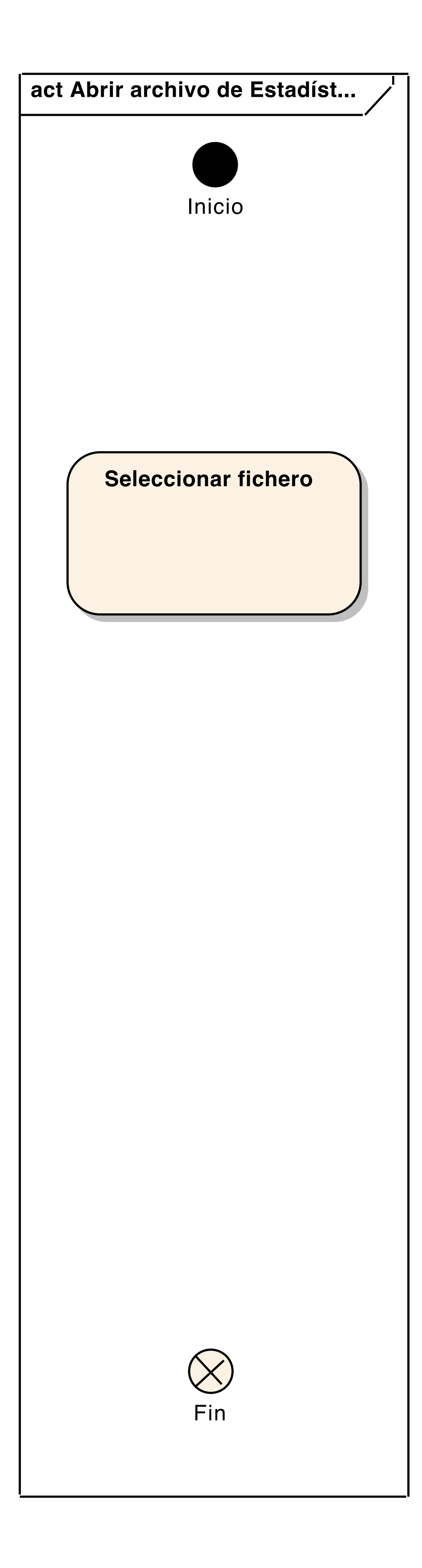


Ilustración 39 – Diagrama de Actividades de Analizador (Abrir archivo de estadísticas)

5.6.5.6. Cargar archivo de Estadísticas por Red

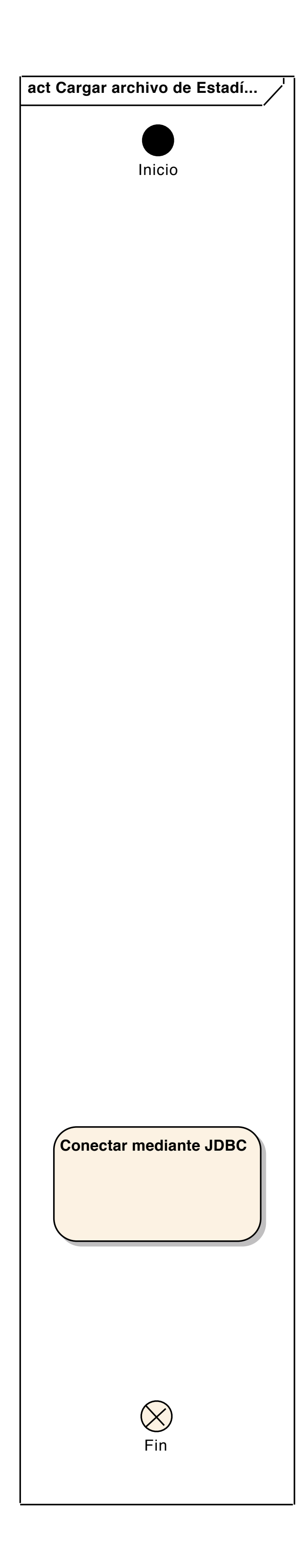


Ilustración 40 – Diagrama de Actividades de Analizador (Obtener estadísticas por red)

5.7. TABLA DE ESTADÍSTICAS

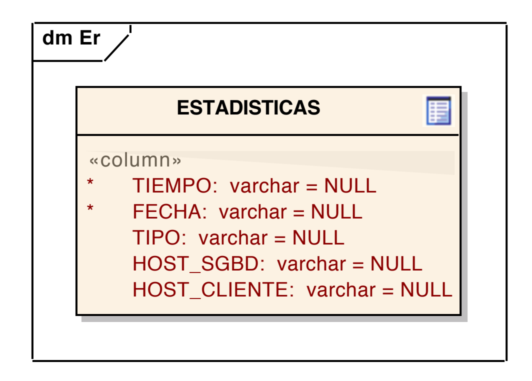


Ilustración 41 – Diagrama de Actividades E/R del sistema de almacenamiento de estadísticas

La base de datos de estadísticas tan solo contiene una tabla. En dicha tabla se almacenan los datos obtenidos por las métricas de tiempos llevadas a cabo por Servicio.

En la tabla se almacena:

- Tiempo. Es el tiempo de latencia de la petición
- Fecha. Fecha y hora de la muestra.
- Tipo. Tipo o Script del que se han tomado las métricas.
- Host_sgbd. Nombre del host que despliega el sistema gestor de bases de datos.
- Host_cliente. Nombre del host cliente que ha lanzado la petición.

5.8. ACLARACIONES SOBRE EL DESARROLLO.

5.8.1. EN REFERENCIA A LA FORMA DE ALMACENAMIENTO DE ESTADÍSTICAS.

Cuando nos referimos a abrir un fichero de estadísticas dentro de la aplicación Analizador, nos referimos a estadísticas almacenadas en un fichero SQLite, ya que este es el formato por defecto para dichos datos.

Si los datos se han guardado en un sistema de bases de datos relacional como puede ser Oracle, SQL Server,…, para obtener los datos a analizar hemos de hacerlo con la opción de recibir fichero por red.

La opción de recibir fichero por red hace una llamada a Servicio para que el nos envíe los datos con los que está trabajando y Analizador se encarga de almacenarlos en el formato nativo de este, es decir, SQLite.

5.8.2. EN REFERENCIA A LOG4J Y SU CAPACIDAD DE ALMACENAR EL DIARIO DE EJECUCIÓN EN UNA TABLA DE UNA BASE DE DATOS MEDIANTE JDBC

Log4j tiene mecanismos para guardar los diarios de ejecución que va redactando en distintos formatos. Por defecto este proyecto viene configurado para almacenar los diarios de ejecución en un fichero de texto. El objetivo de esta nota era aclarar porque no se emplea un fichero SQLite para almacenar los diarios de ejecución. La respuesta es simple. Un requisito de todas las aplicaciones que forman este proyecto es el bajo consumo de recursos. Emplear la conexión JDBC con Log4J aumenta el consumo de

recursos, además de que puede ser un foco de errores innecesarios. Por eso se opto por un fichero de texto. Hay que decir que todo esto es configurable sin que afecte al código, luego si se quiere se pude hacer.

5.8.3. En referencia a la existencia de dos tipos de lanzadores de ejecución de tareas

Para la definición de tareas se pide un alias. Este alias distingue tareas de ejecución de scripts SQL y procesos de PL/SQL o T-SQL.

¿Por qué se hace esto? Pues muy fácil, para ejecutar código PL/SQL o T-SQL se hacen verificaciones para evitar fallos a la hora de lanzarlos ya que la latencia de recuperación de fallos a la hora de lanzar tareas que efectúen procesos es mayor que la de sentencias SQL.

Las hebras de ejecución de sentencias SQL son más livianas que las que ejecutan procesos PL/SQL o T-SQL, ya que no hacen verificaciones. Para este tipo de ejecuciones es más costoso intentar predecir el fallo, que dejarlo ejecutarse.

Una hebra PL/SQL o T-SQL, puede ejecutar perfectamente scripts que solo contenga código de sentencias SQL lo que ocurre es que su ejecución suele ser algo más lenta.

6. CALIDAD DEL SOFTWARE DESARROLLADO

Todas las metodologías y herramientas tienen un único fin producir software de gran calidad.

¿Cómo podemos definir la calidad del software?

"Concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos con los estándares de desarrollo explícitamente documentados y con las características implícitas que se espera de todo software desarrollado profesionalmente" R. S. Pressman (1992).

"El conjunto de características de una entidad que le confieren su aptitud para satisfacer las necesidades expresadas y las implícitas" ISO 8402 (UNE 66-001-92).

Luego resumiendo, los requisitos del software son la base de las medidas de calidad, la falta de concordancia con los requisitos es una falta de calidad.

Los estándares o metodologías definen un conjunto de criterios de desarrollo que guían la forma en que se aplica la ingeniería del software. Si no se sigue ninguna metodología siempre habrá falta de calidad

Existen algunos requisitos implícitos que a menudo no se mencionan, que también pueden implicar una falta de calidad. Estos requisitos son tan importantes como planificar un mantenimiento, generar código estándar que facilite el mantenimiento. La verificación formal del mismo, o por lo menos la verificación formal estática.

Cómo intentar asegurar la calidad del software

La garantía de calidad del software es el conjunto de actividades planificadas y sistemáticas necesarias para aportar los mecanismos de confianza en que el producto software satisfará los requisitos dados de calidad.

Los planteamientos de calidad del software es un proceso previo y propio de cada aplicación antes de comenzar a desarrollarla y no después.

La garantía de calidad del software está presente en:

- Los métodos y herramientas de análisis, diseño, programación y prueba. En el caso de este proyecto en cuestión, se ha empleado UML y Desarrollo Iterativo Incremental. Para ello se ha empleado Enterprise Architect como apoyo a este desarrollo.

- Inspecciones técnicas formales en todos los pasos del proceso de desarrollo del software. Para esto se ha empleado PMD y Checkstyle. Son herramientas muy extendidas a la hora de la verificación formal de código Java. PMD sirve para

- Estrategias de prueba parcial y total. JUnit y pruebas parciales de prototipos. JUnit es un conjunto de clases que permite realizar la ejecución de clases Java de manera controlada, para poder evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera. Control de la documentación del software y de los cambios realizados. Durante el desarrollo se ha empleado Subversion.

- Procedimientos para ajustarse a los estándares y dejar claro cuando se está fuera de ellos. Esto no es aplicable a este proyecto, ya que es plenamente estándar.

- Mecanismos de medida o métricas.

- Registro de auditorías y realización de informes.

Actividades para asegurar la garantía de calidad del software:

- Métricas de software para el control del proyecto.

- Verificación y validación del software a lo largo del ciclo de vida.

Incluye las pruebas y los procesos de revisión e inspección y la gestión de la configuración del software.

Gestión de la calidad del software, estándares internacionales

Aquí damos un vistazo a los estándares internacionales que están orientados a prácticas empresariales. No es aplicable a este proyecto, pero tiene una utilidad didáctica.

Gestión de la calidad (ISO 9000). Conjunto de actividades de la función general de la dirección que determina la calidad, los objetivos y las responsabilidades y se implanta por medios tales como la planificación de la calidad, el control de la calidad, el garantía de la calidad y la mejora de la calidad, en el marco del sistema de calidad.

Política de calidad (ISO 9000). Directrices y objetivos generales de una organización, directrices y objetivos a la calidad, tal como se expresan formalmente por la alta dirección.

La gestión de la calidad se aplica normalmente a nivel de empresa.

También puede haber una gestión de calidad dentro de la gestión de cada proyecto.

Control de la calidad del software

Son las técnicas y actividades de carácter operativo, utilizadas para satisfacer los requisitos relativos a la calidad, centradas en dos objetivos fundamentales:

- Mantener bajo control un proceso.
- Eliminar las causas de los defectos en las diferentes fases del ciclo de vida

En general son las actividades para evaluar la calidad de los productos desarrollados.

Sistema de calidad

Los sistema de calidad están formados por una estructura organizativa, unos procedimientos para garantizar la calidad y un conjunto de procesos y recursos necesarios para implantar la gestión de calidad.

El sistema de calidad se debe adecuar a los objetivos de calidad de la empresa. Son políticas globales.

La dirección de la empresa es la responsable de fijar la política de calidad y las decisiones relativas a iniciar, desarrollar, implantar y actualizar el sistema de calidad.

Un sistema de calidad consta de varias partes:

- Documentación.
- Manual de calidad. Es el documento principal para establecer e implantar un sistema de calidad. Puede haber manuales a nivel de empresa, departamento, producto, específicos como para compras, proyectos,…
- Parte física: locales, herramientas ordenadores,…
- Aspectos humanos:
  - Formación de personal.
  - Creación y coordinación de equipos de trabajo.
- Normativas para garantizar la calidad:
  - ISO 9000: Gestión y aseguramiento de calidad, son conceptos y directrices generales.
  - Recomendaciones externas para aseguramiento de la calidad (ISO 9001, ISO 9002, ISO 9003). Son extensiones de la anterior.

- Recomendaciones internas para aseguramiento de la calidad (ISO 9004). Un extra.

Factores que determinan la calidad del software

Se clasifican en tres grupos:

- Operaciones del producto: características operativas.
  - Corrección. ¿Hace lo que se le pide? El grado en que una aplicación satisface sus especificaciones y consigue los objetivos encomendados por el cliente.
  - Fiabilidad. ¿Lo hace de forma fiable todo el tiempo? El grado que se puede esperar de una aplicación lleve a cabo las operaciones especificadas y con la precisión requerida.
  - Eficiencia. ¿Qué recursos hardware y software necesito? La cantidad de recursos hardware y software que necesita una aplicación para realizar las operaciones con los tiempos de respuesta adecuados.
  - Integridad. ¿Puedo controlar su uso? El grado con que puede controlarse el acceso al software o a los datos a personal no autorizado.
  - Facilidad de uso. ¿Es fácil y cómodo de manejar? El esfuerzo requerido para aprender el manejo de una aplicación, trabajar con ella, introducir datos y conseguir resultados.
- Revisión del producto. Capacidad para soportar cambios. Su extensibilidad.
  - Facilidad de mantenimiento. ¿Puedo localizar los fallos? El esfuerzo requerido para localizar y reparar errores.

- Flexibilidad. ¿Puedo añadir nuevas opciones? El esfuerzo requerido para modificar una aplicación en funcionamiento.

- Facilidad de prueba. ¿Puedo probar todas las opciones? El esfuerzo requerido para probar una aplicación de forma que cumpla con lo especificado en los requisitos.

- Transición del producto: adaptabilidad a nuevos entornos

  - Portabilidad. ¿Podré usarlo en otra máquina? El esfuerzo requerido para transferir la aplicación a otro hardware o sistema operativo.

  - Reusabilidad. ¿Podré utilizar alguna parte del software en otra aplicación? Grado en que partes de una aplicación pueden utilizarse en otras aplicaciones.

  - Interoperabilidad. (¿Podrá comunicarse con otras aplicaciones o sistemas informáticos? El esfuerzo necesario para comunicar la aplicación con otras aplicaciones o sistemas informáticos.

Métricas de la calidad del sofware

Es difícil, y en algunos casos imposibles, desarrollar medidas directas de los factores de calidad del software. Cada factor de calidad Fc se puede obtener como combinación de una o varias métricas:

$$Fc = \sum_{i=1}^{n} c_i * m_i$$

Siendo c_i factor de ponderación de la métrica i, que dependerá de cada aplicación específica y m_i métrica i. Habitualmente se puntúan de 0 a 10 en las métricas y en los factores de calidad.

Métricas para determinar los factores de calidad:

- Facilidad de auditoría.
- Exactitud.
- Normalización de las comunicaciones.
- Completitud.
- Concisión.
- Consistencia.
- Estandarización de los datos.
- Tolerancia de errores.
- Eficiencia de la ejecución.
- Facilidad de expansión.
- Generalidad.
- Independencia del hardware.
- Instrumentación.
- Modularidad.
- Facilidad de operación.
- Seguridad.
- Autodocuemntación.
- Simplicidad.

- Independencia del sistema software.
- Facilidad de traza.
- Formación.

Autoevaluación

Los pilares básicos de la certificación de calidad del software son:

- Una metodología adecuada.
- Un medio de valoración de la metodología.
- Un reconocimiento de la industria de la metodología utilizada y del medio de valorar la metodología.
- Todas las afirmaciones anteriores son correctas.
- Ninguna respuesta anterior es correcta.

La calidad del software implica:

- La concordancia entre el software diseñado y los requisitos.
- Seguir un estándar o metodología en el proceso de desarrollo de software.
- Tener en cuenta los requisitos implícitos, no expresados por los usuarios.
- Todas las afirmaciones anteriores son correctas.
- Ninguna respuesta anterior es correcta.

7. CONCLUSIONES Y LÍNEAS FUTURAS

El proyecto se basa en un estudio previo desarrollado en la asignatura de Laboratorio de Bases de Datos, del que en la actualidad no comparte ni una línea de código, y en el que se ha intentado mejorar los resultados obtenidos y la forma de obtenerlo.

El proyecto se ha realizado de una forma madura y organizada muy centrada en el desarrollo rápido de soluciones y sobre las que se han ido aportando mejoras, partiendo siempre de desarrollo de prototipos para que siempre se puede pueda ver los cambios en entre los distintos prototipos. Partiendo obviamente de la compatibilidad entre ellos, y siempre añadiéndoles funcionalidades pero manteniendo las ya implementadas, todo esto se consigue partiendo de un buen análisis inicial.

En cuanto al desarrollo, indicar que para el desarrollo especialmente de la interfaz gráfica he necesitado un proceso de formación de la tecnología empleada para el desarrollo de la interfaz gráfica, en especial de las bibliotecas de JFreeChart.

Líneas futuras

En cuanto las posibles continuaciones o ampliaciones del proyecto, para empezar, he de indicar que JFreeChart se ha empleado ya que es gratuito y se distribuye con licencia LGPL, pero el resultado gráfico es mejorable. Luego este componente sería un candidato a ser sustituido por otro.

En línea general, los componentes tienen una arquitectura extensible. Por aplicaciones, aquí se enumeran algunas posibles líneas de trabajo:

- Servicio:
    - Diseñar un sistema inteligente que cree script SQL y/o PL/SQL con datos aleatorios.

- Estudiar la posibilidad de crear un sistema de predicción de tiempos. De modo que si se supera las cotas máximas, lo indique de alguna forma. Esto se desarrollaría con una red neuronal que fuera capaz de aprender ha interpretar los resultados.

- Sustituir el driver JDBC de SQLite por otro, ya que a veces da fallos bajo Linux. En Linux, si no existe el fichero de estadísticas de SQLite, no es capaz de crearlo hay que crear el fichero vacio con la estructura de datos antes de ejecutar Servicio.

- Analizador:

  - Creación de un sistema de cargado de complementos y extensiones que se pudieran cargar dinámicamente. Ampliando así sus funcionalidades.

  - Agregarle un sistema inteligente que analice un sistema gestor de bases de datos, y no haga un planteamiento actual de la salud del mismo, y en caso de que encuentre algún problema, nos indique el origen y nos ofrezca soluciones.

8. APÉNDICE 1: MANUAL DE USUARIO

8.1. MANUAL DE INSTALACIÓN DE SERVICIO

El instalador que se va a mostrar es la versión para Windows. Pero el sistema de instalación es multisistema y está disponible también para entornos Linux.

El proceso de instalación es guiado y ha sido creado gracias a Excelsior Installer. Para más información http://installer.excelsior-usa.com/en/

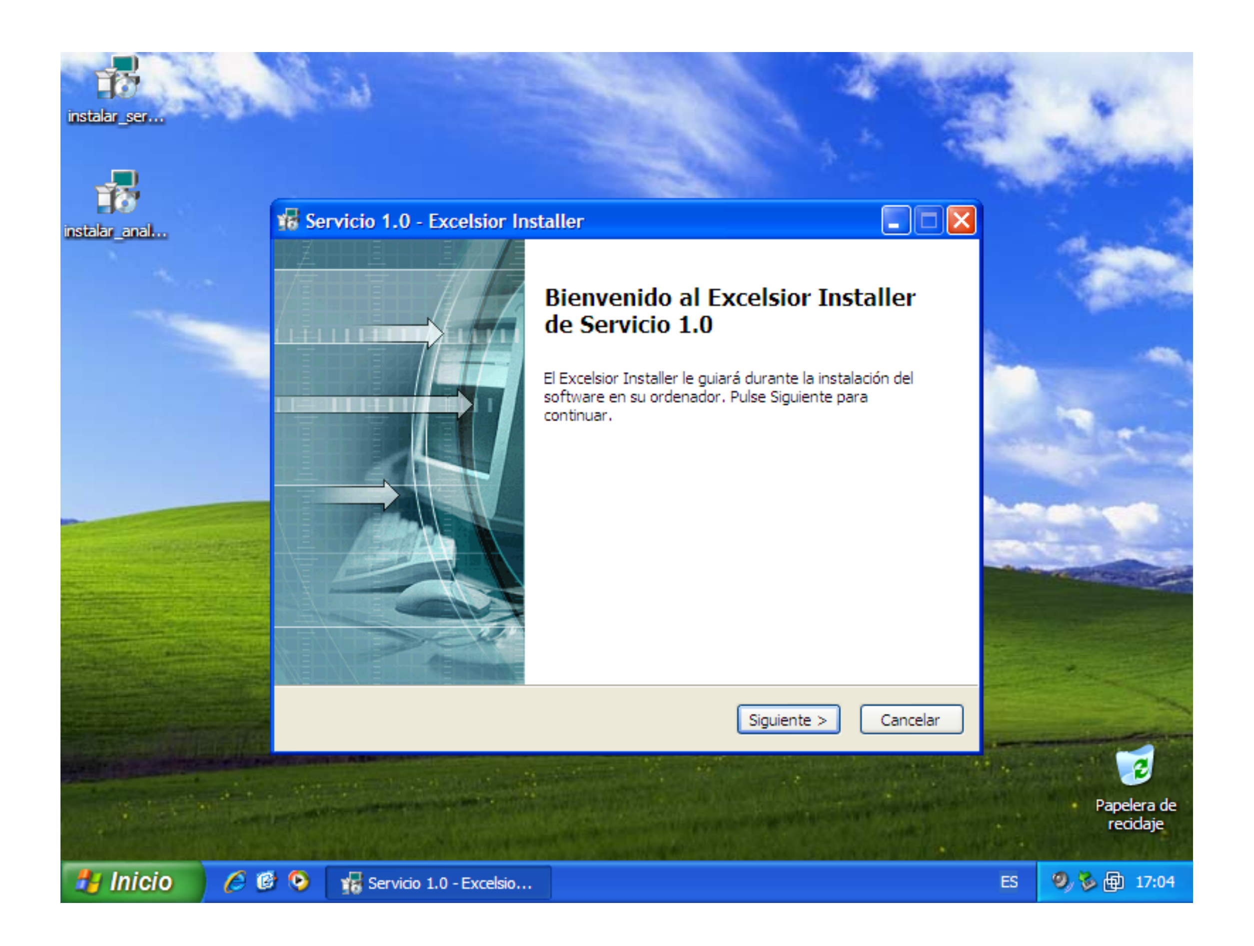


Ilustración 42 – Paso de instalación de Servicio (1)

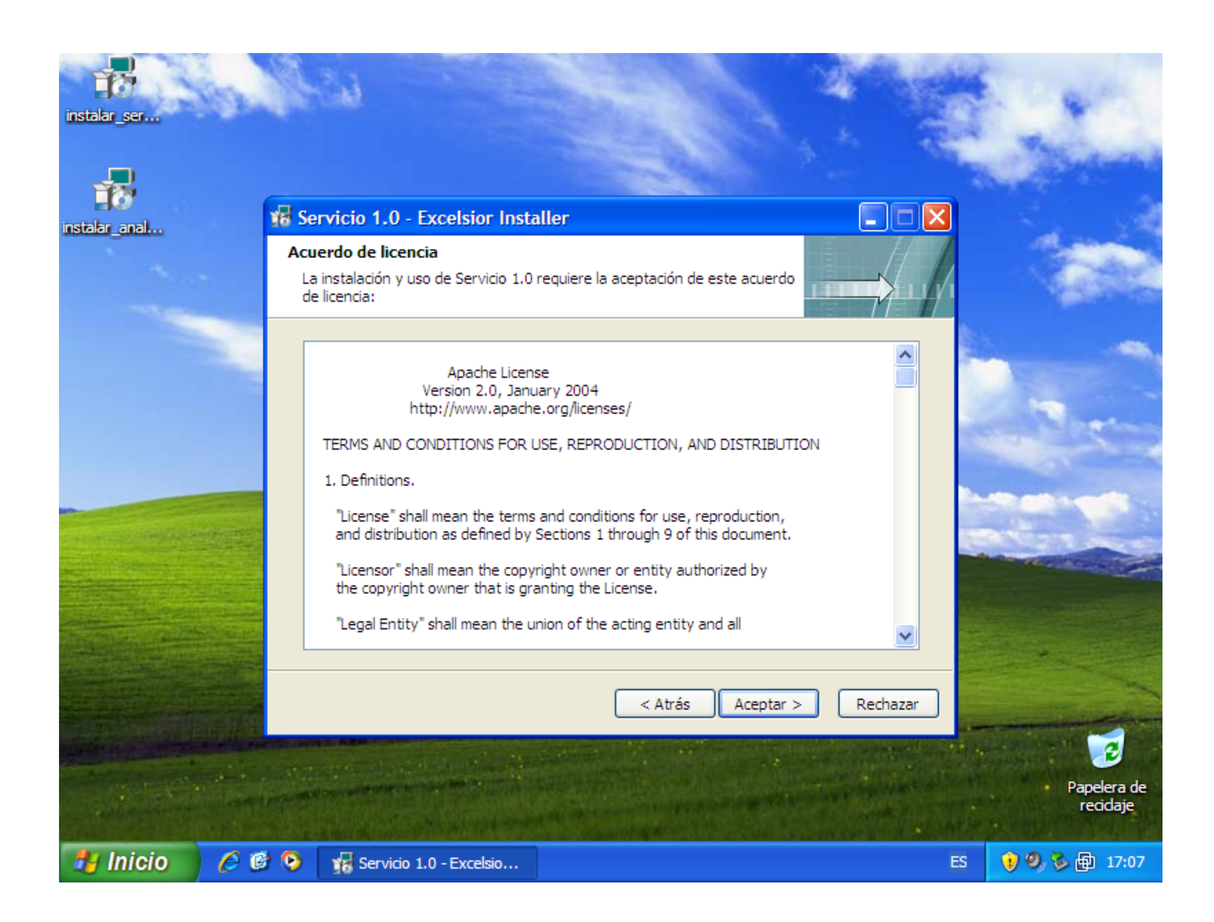


Ilustración 43– Paso de instalación de Servicio (2)

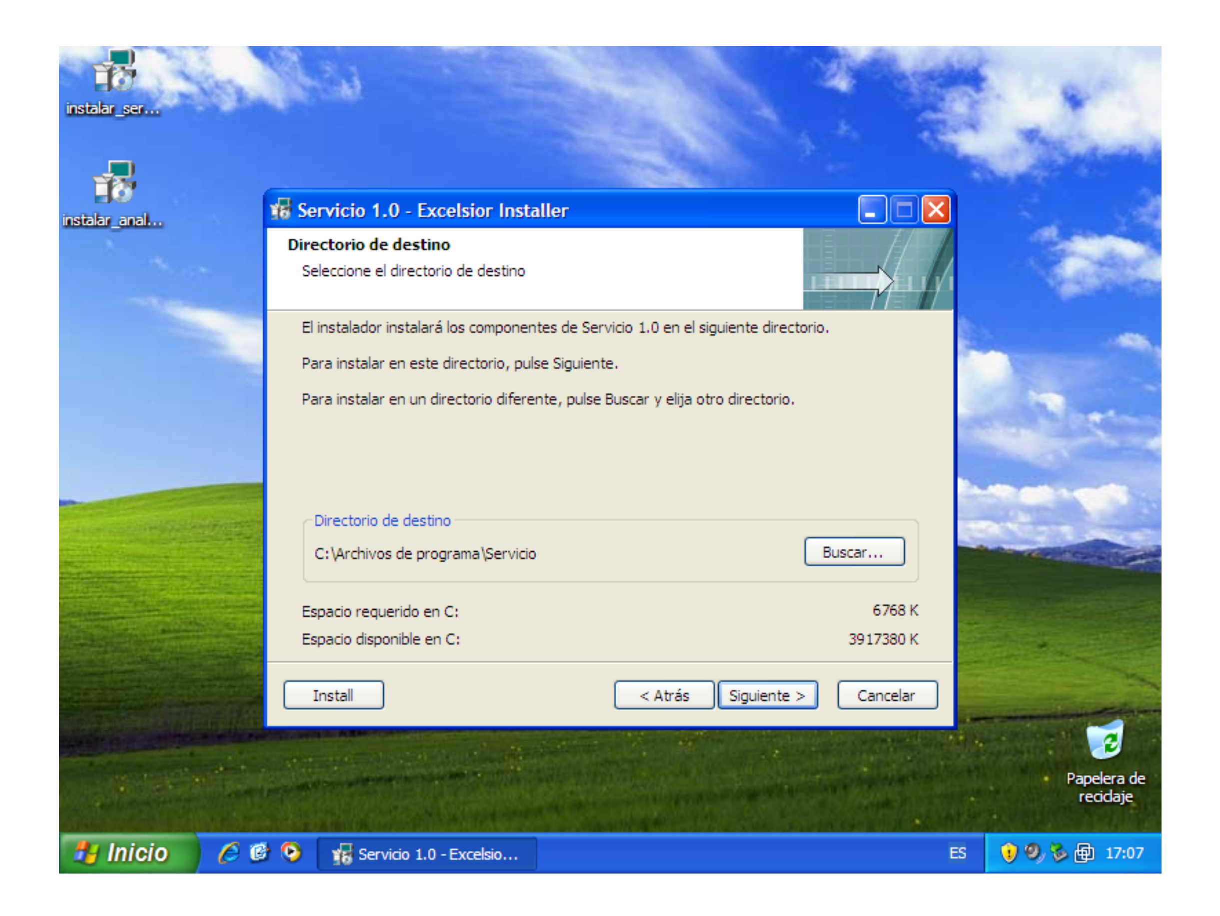


Ilustración 44– Paso de instalación de Servicio (3)

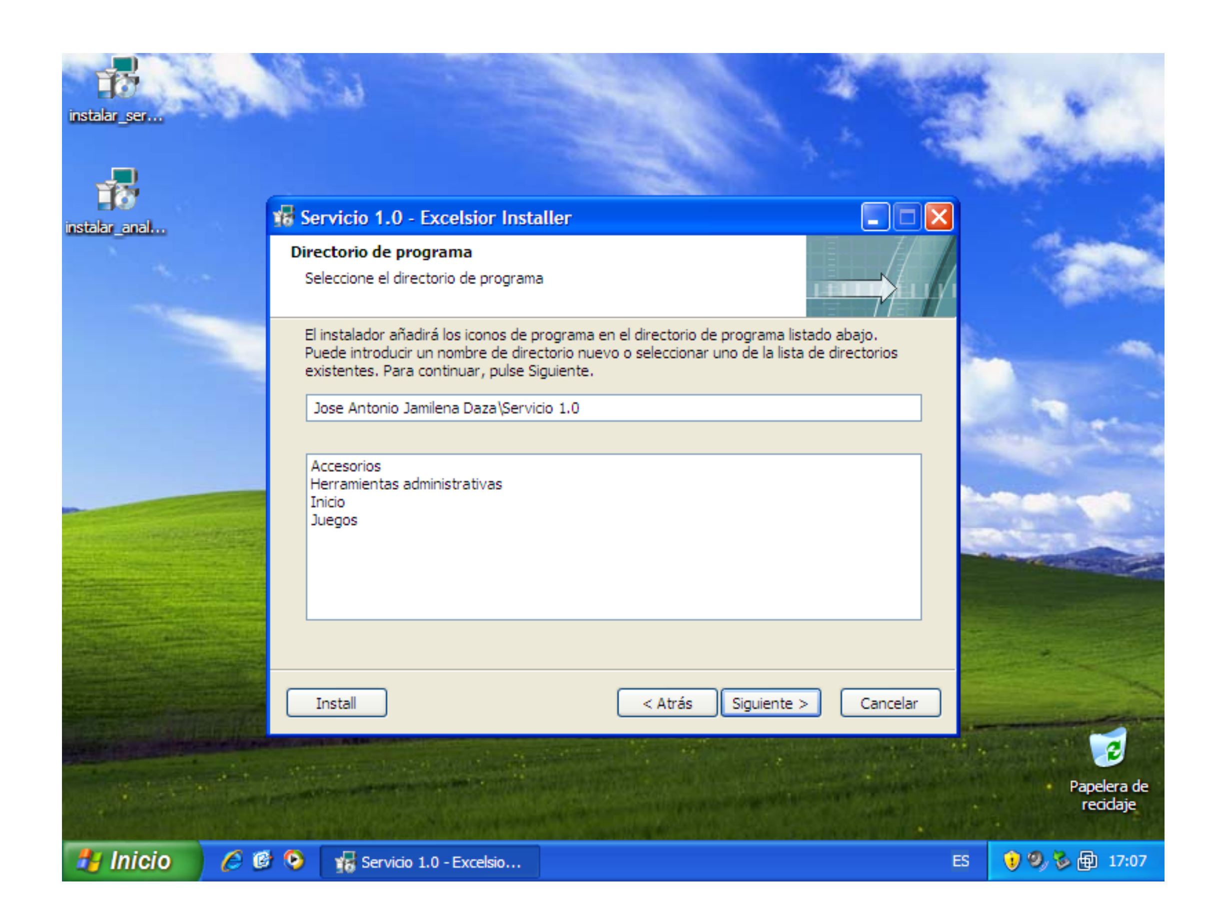


Ilustración 45– Paso de instalación de Servicio (4)

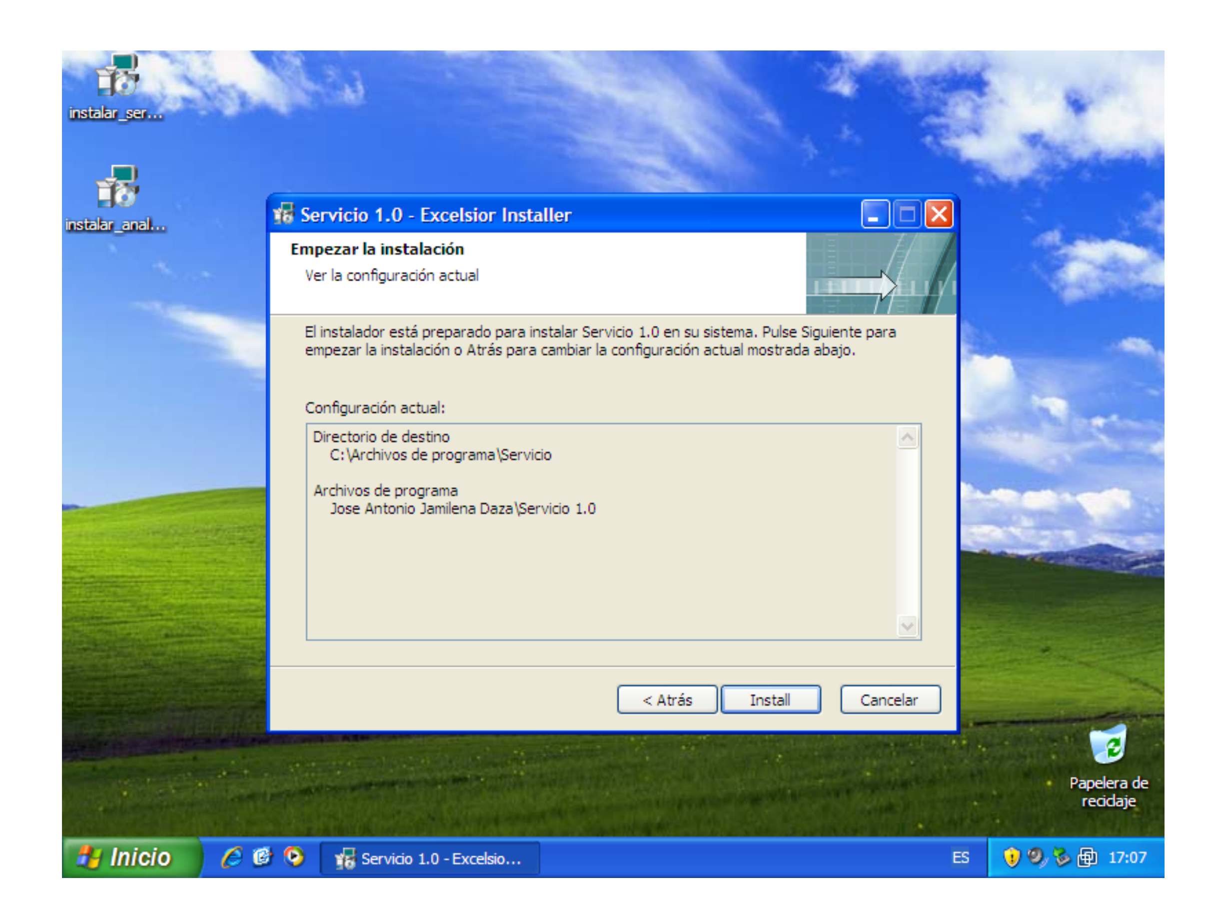


Ilustración 46– Paso de instalación de Servicio (5)

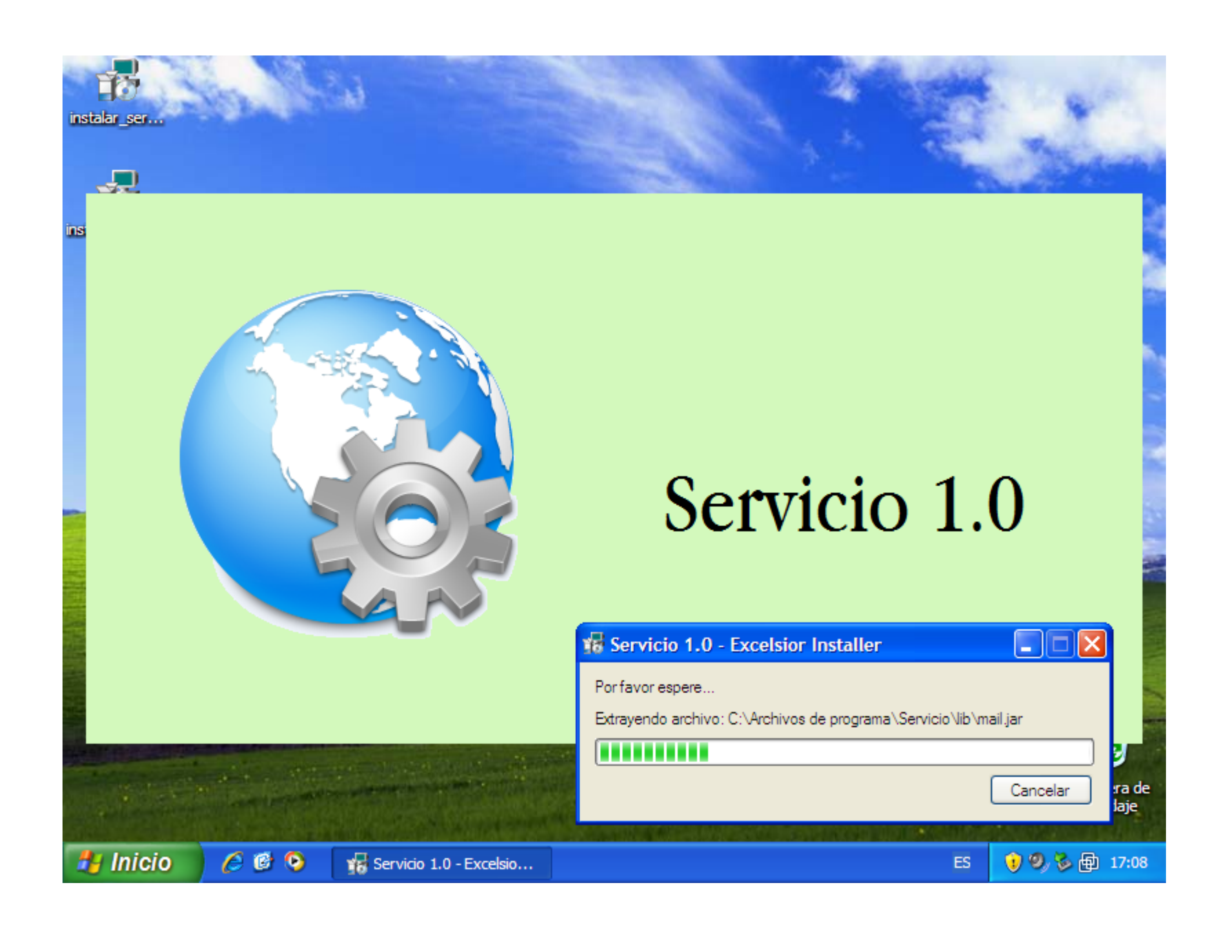


Ilustración 47– Paso de instalación de Servicio (6)

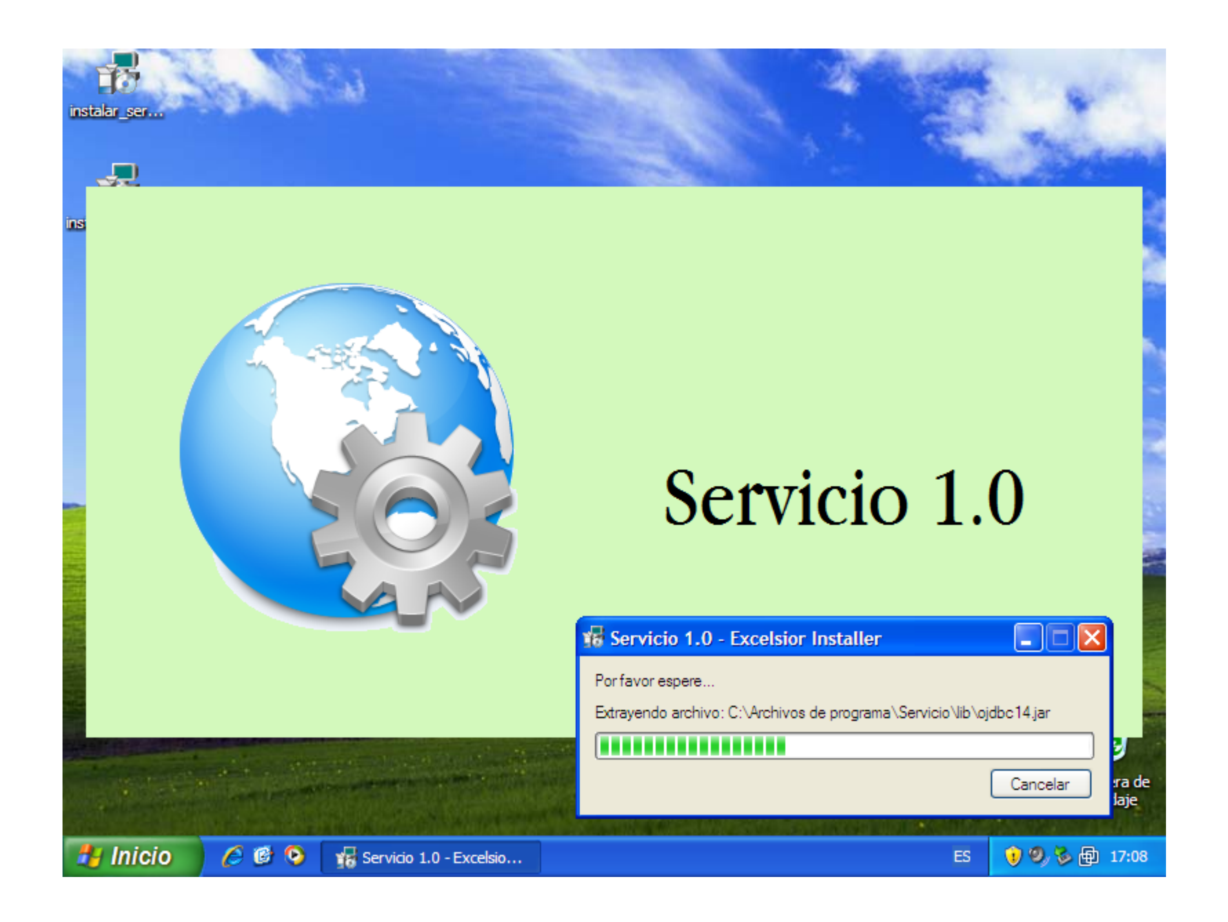


Ilustración 48– Paso de instalación de Servicio (7)

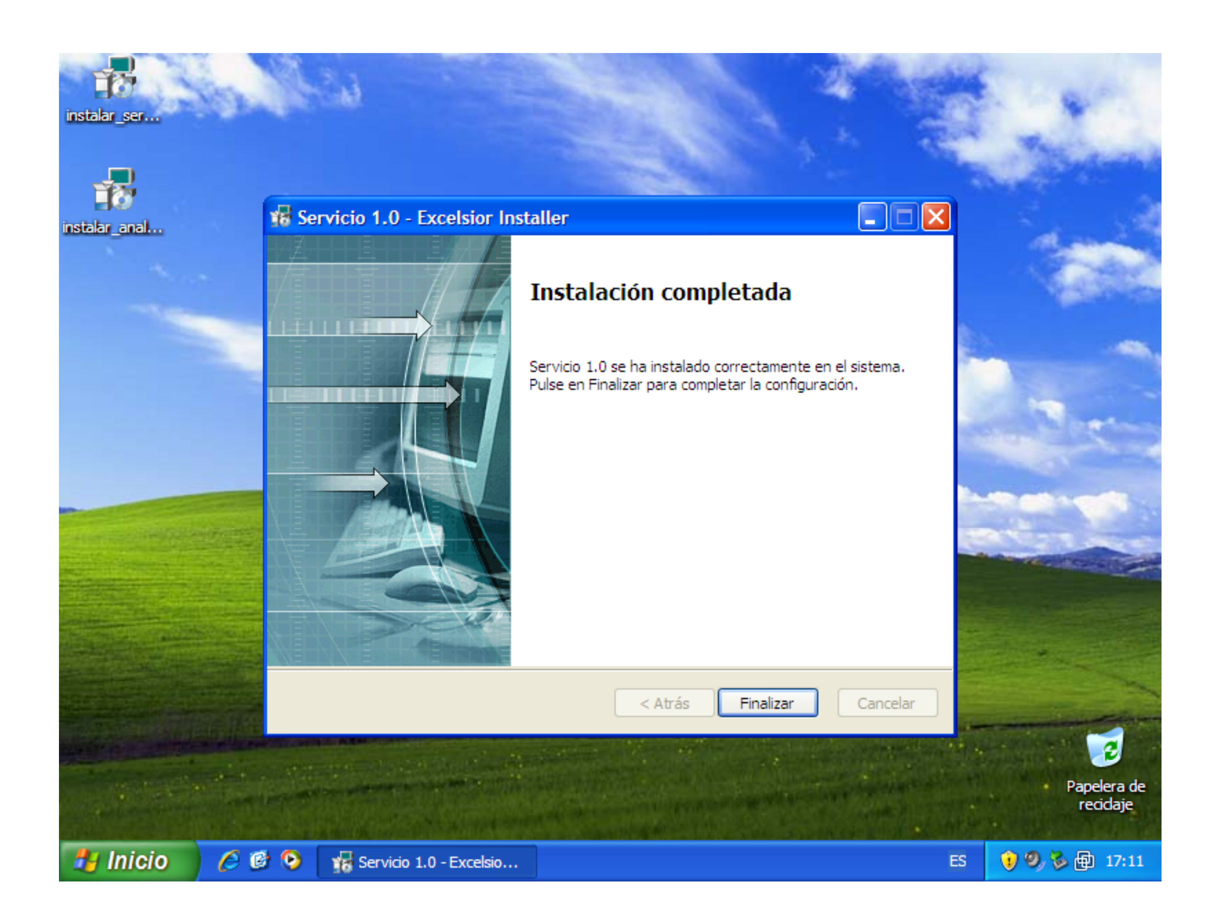


Ilustración 49– Paso de instalación de Servicio (8)

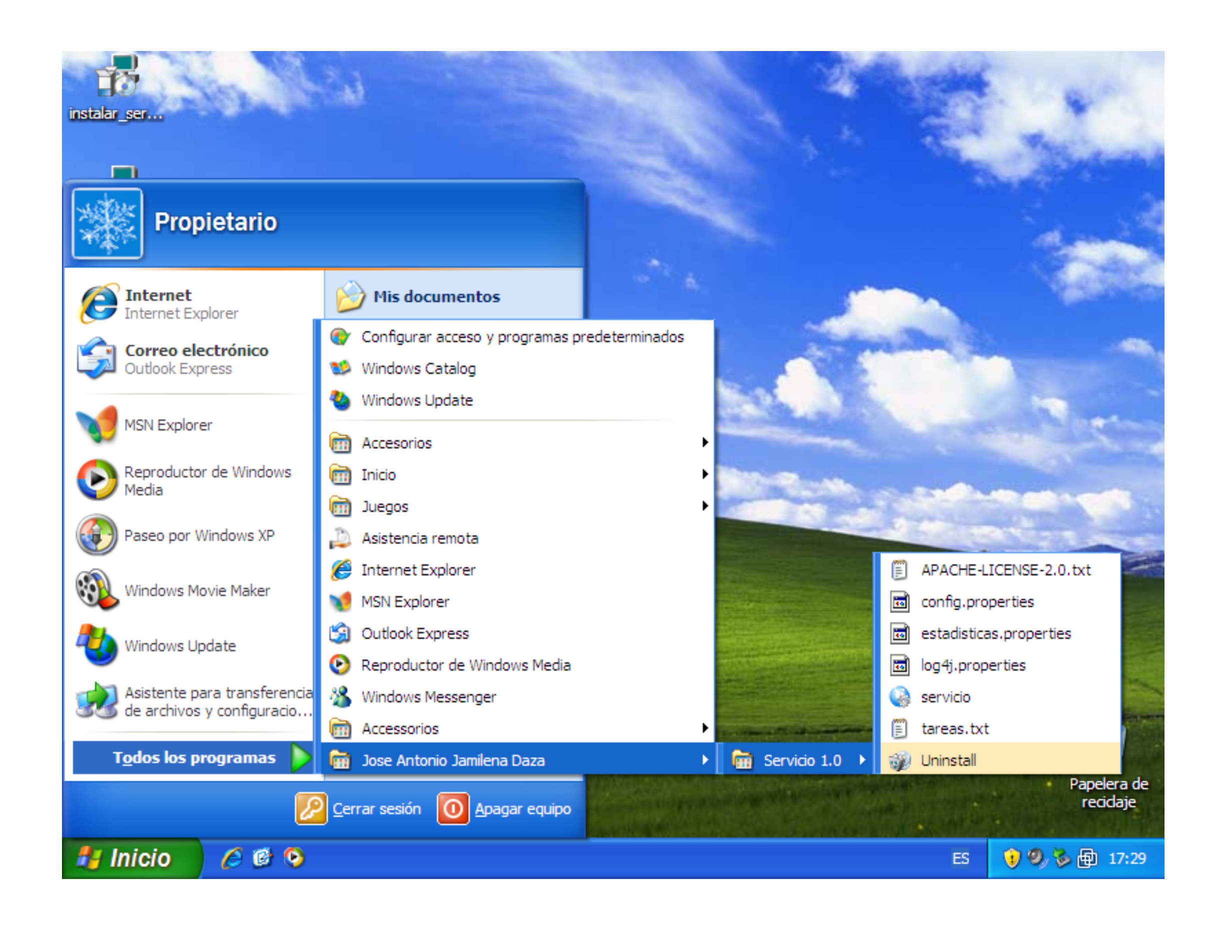


Ilustración 50– Paso de instalación de Servicio (9)

8.2. MANUAL DE INSTALACIÓN DE ANALIZADOR

El instalador que se va a mostrar es la versión para Windows. Pero el sistema de instalación es multisistema y está disponible también para entornos Linux.

El proceso de instalación es guiado y ha sido creado gracias a Excelsior Installer. Para más información http://installer.excelsior-usa.com/en/

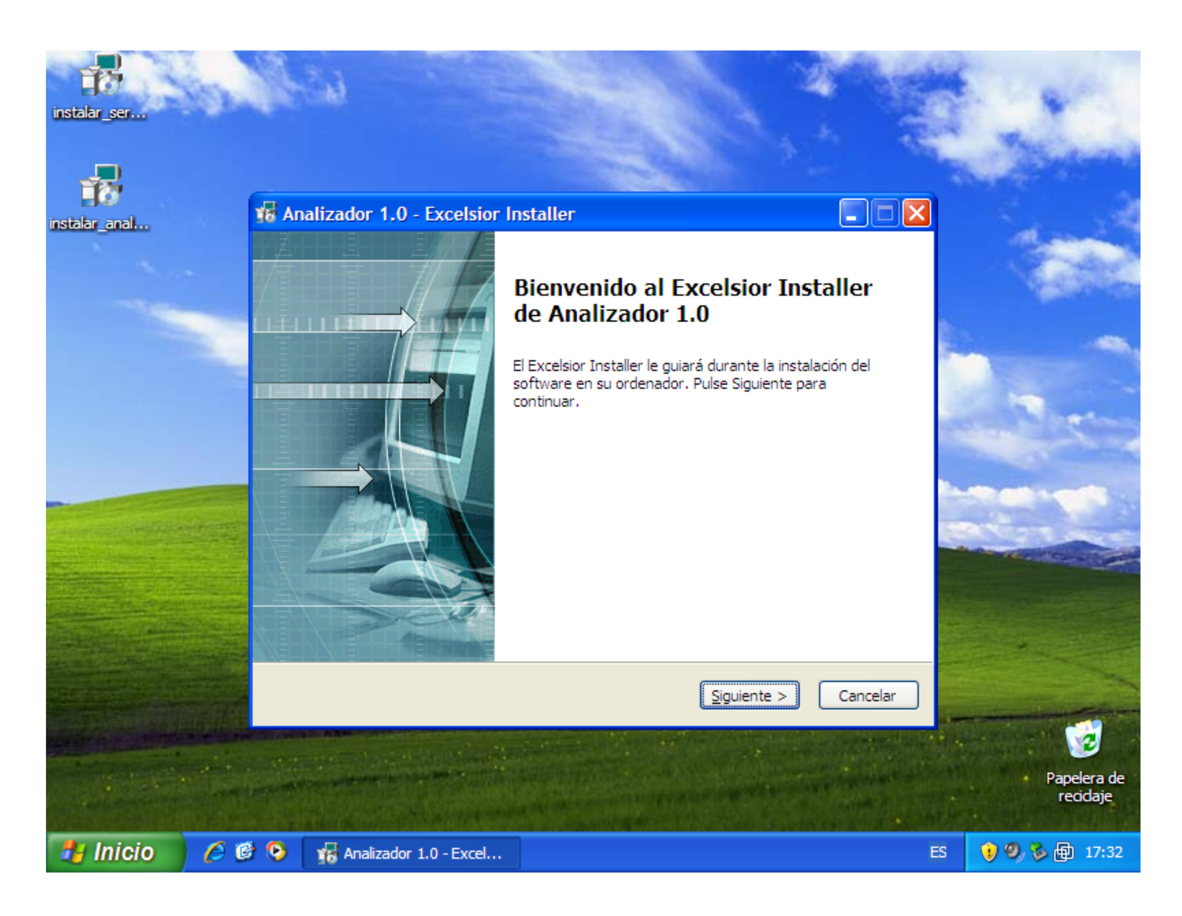


Ilustración 51– Paso de instalación de Analizador (1)

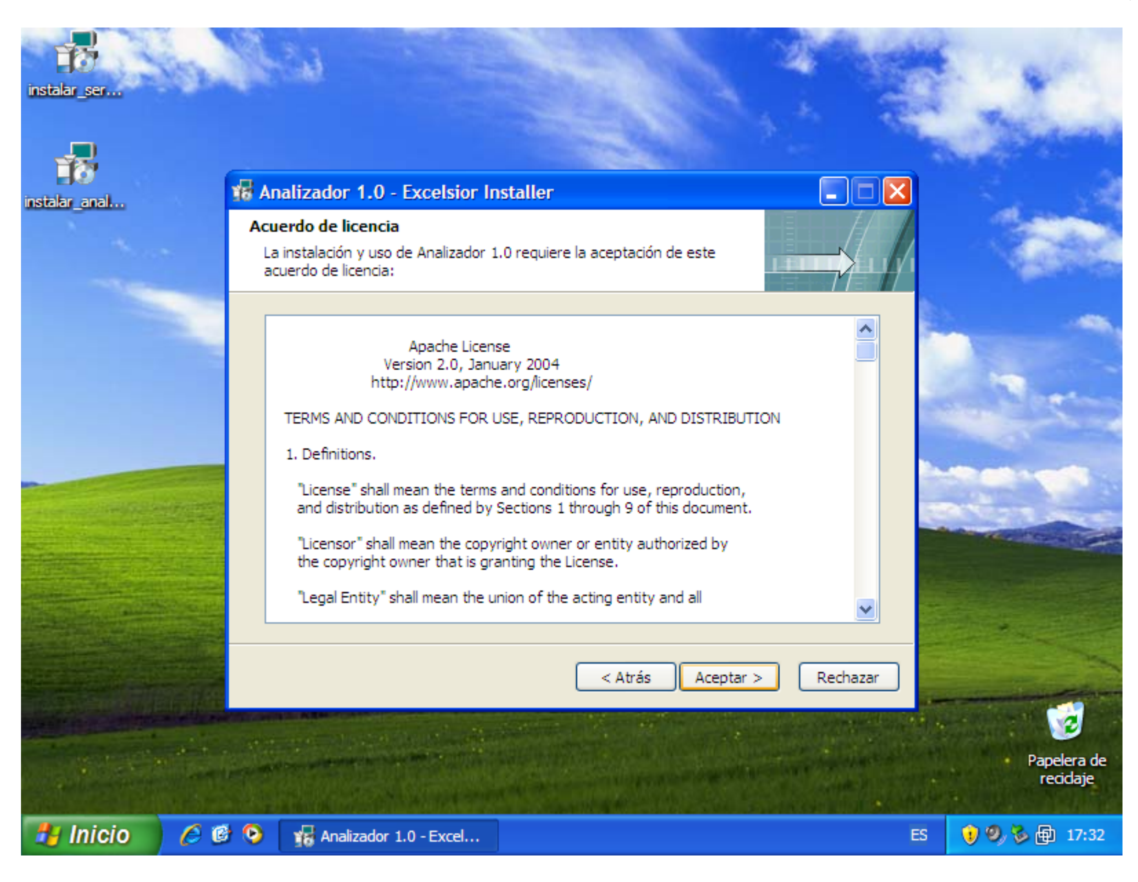


Ilustración 52– Paso de instalación de Analizador (2)

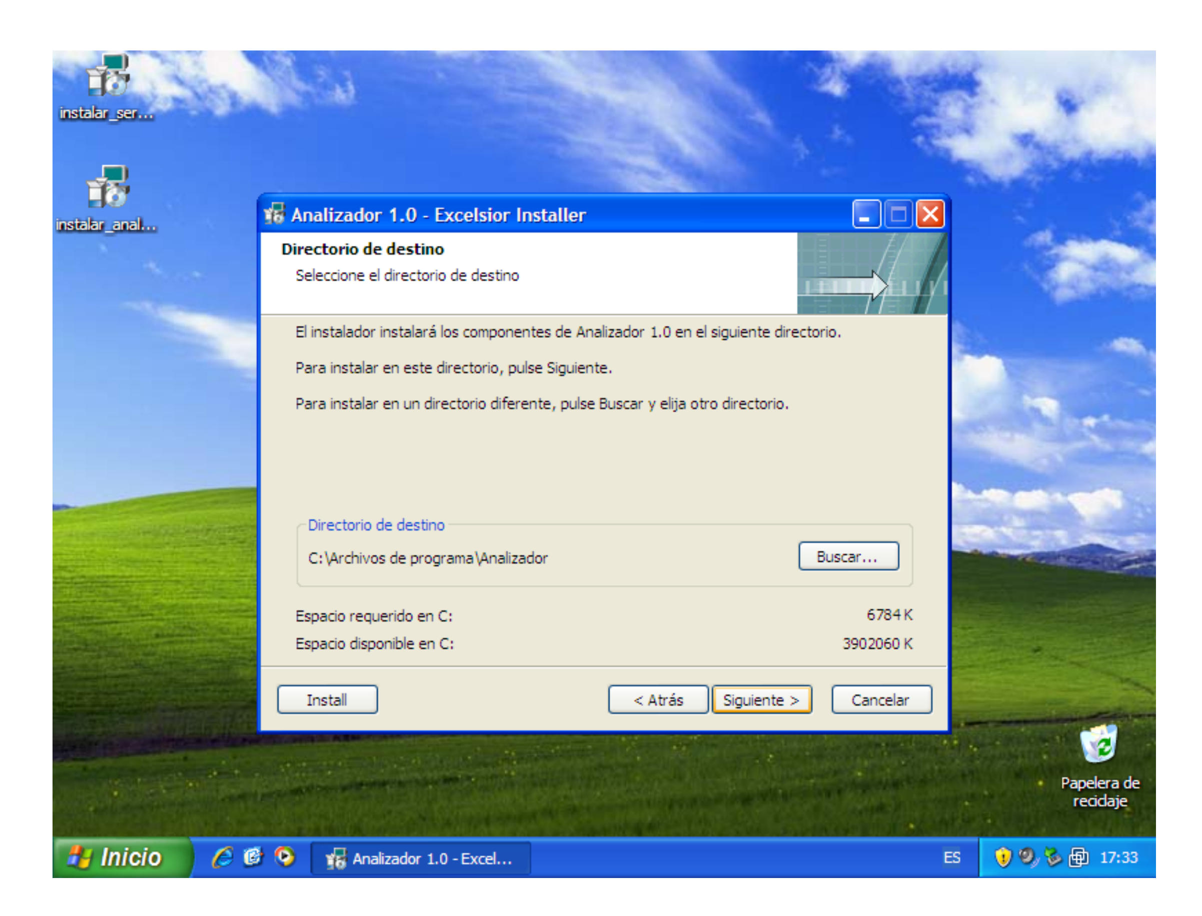


Ilustración 53– Paso de instalación de Analizador (3)

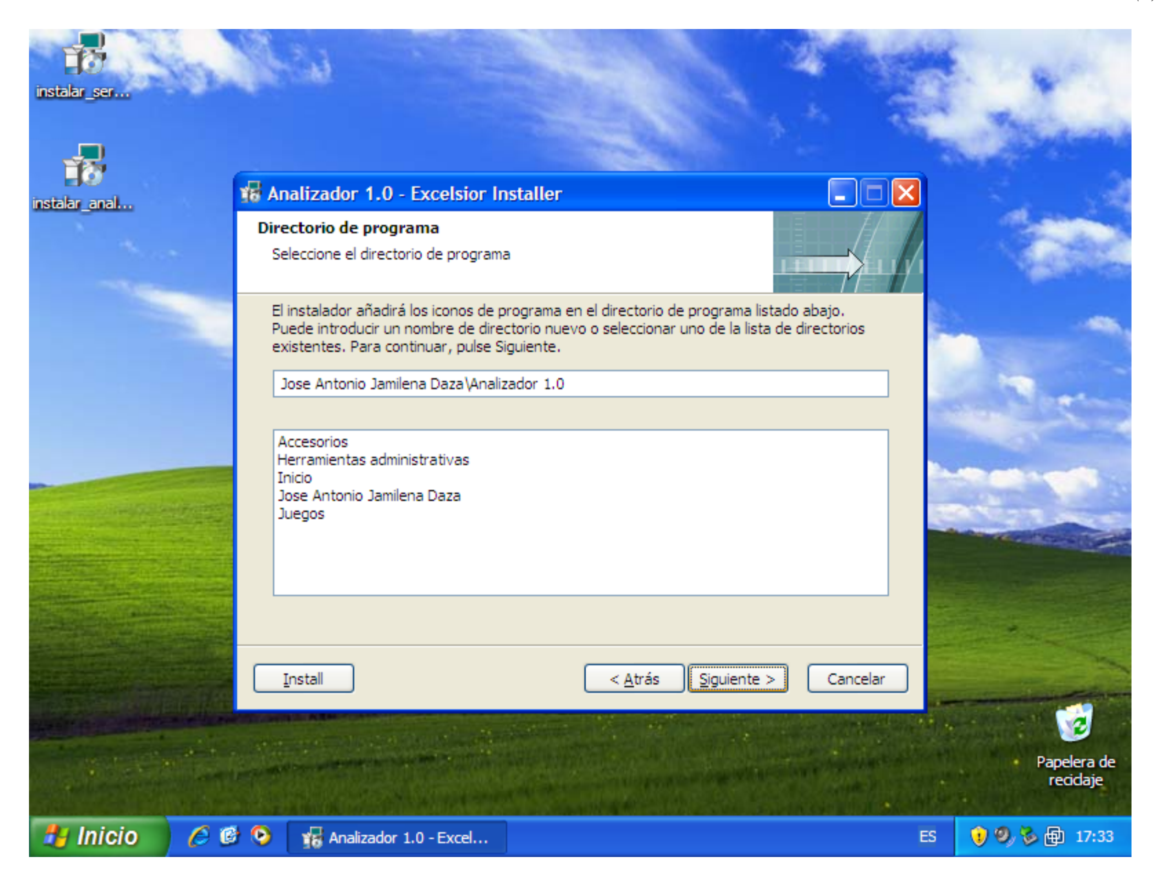


Ilustración 54– Paso de instalación de Analizador (4)

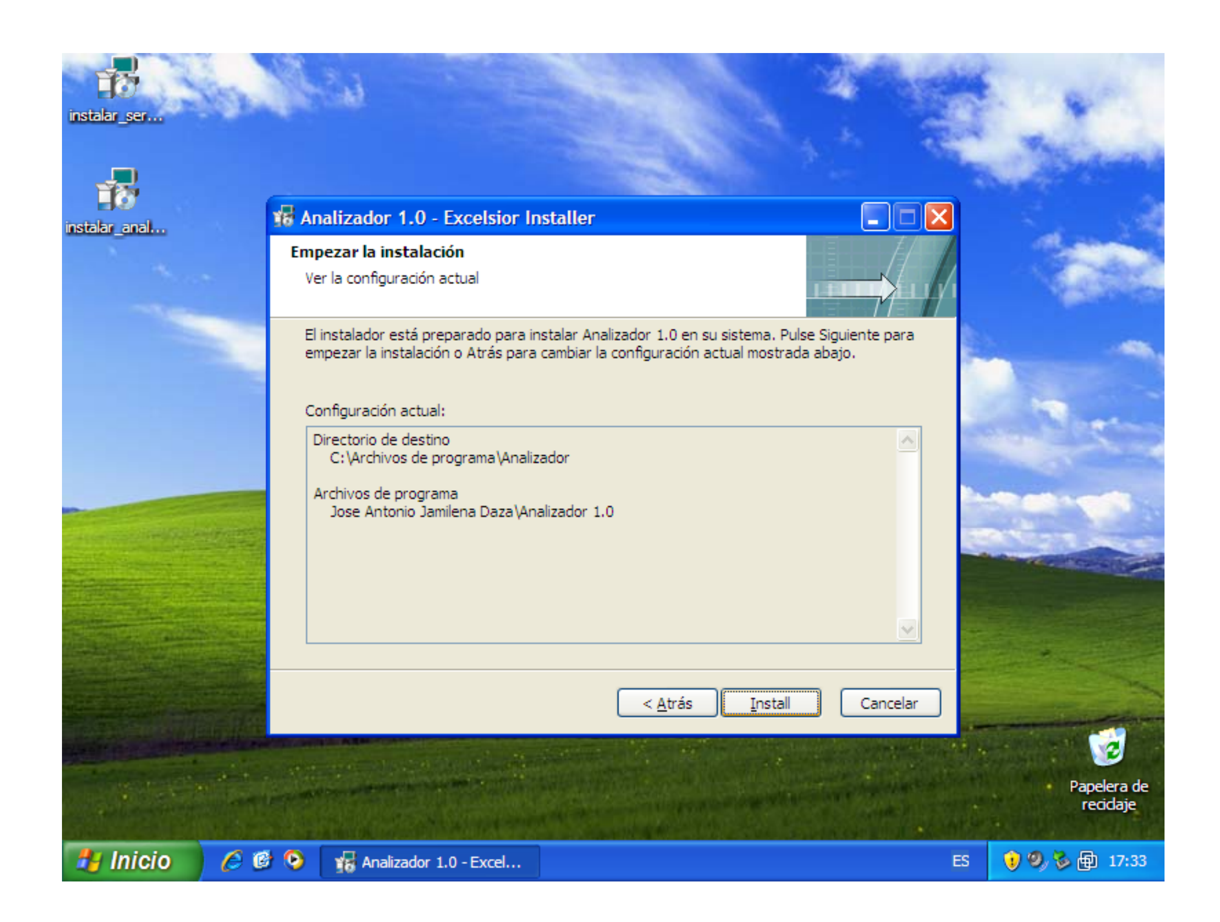


Ilustración 55– Paso de instalación de Analizador (5)

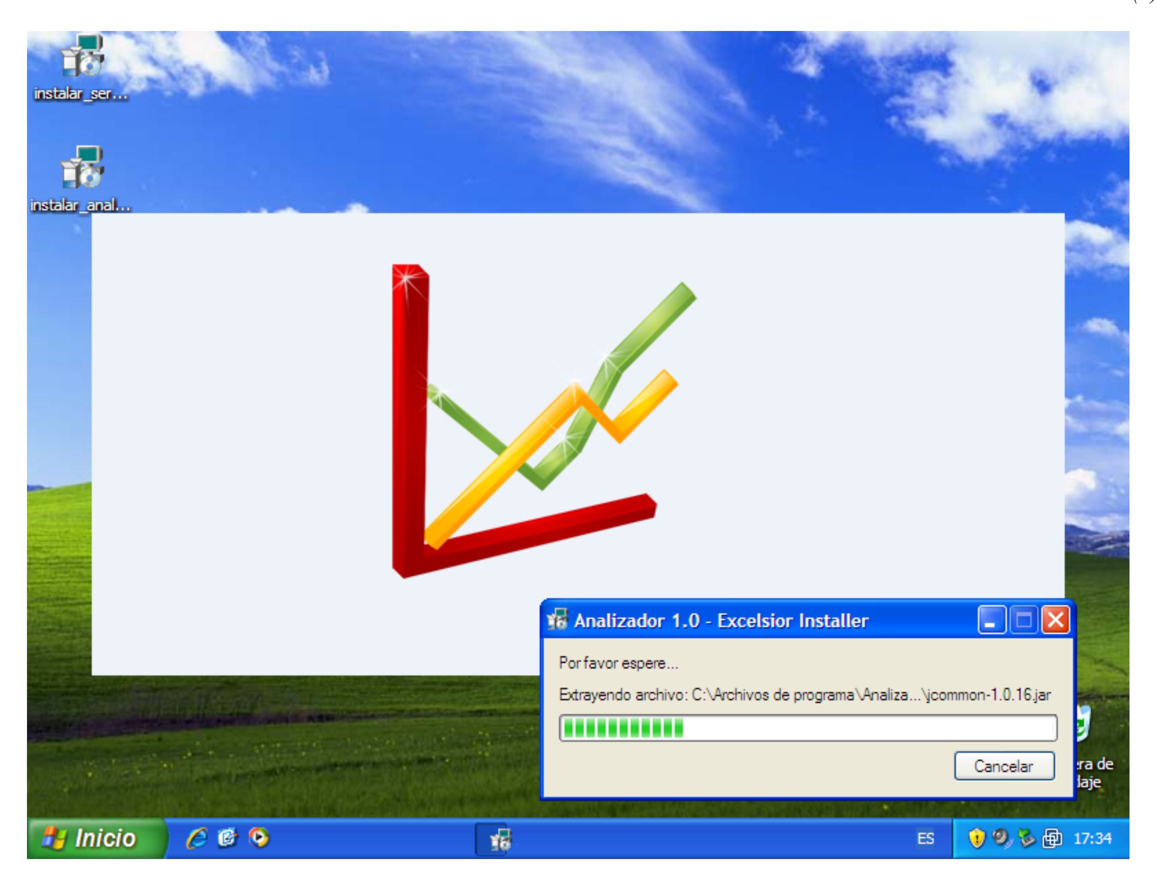


Ilustración 56– Paso de instalación de Analizador (6)

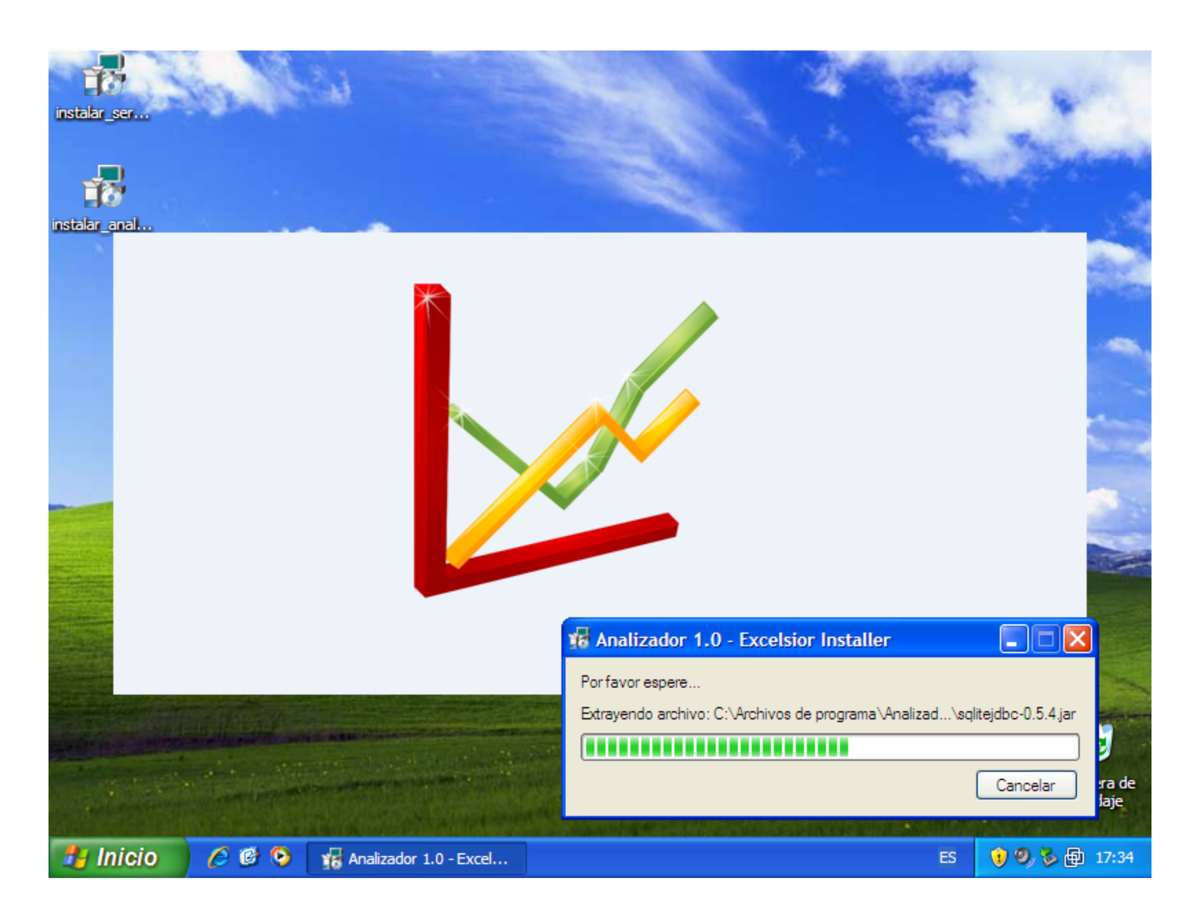


Ilustración 57– Paso de instalación de Analizador (7)

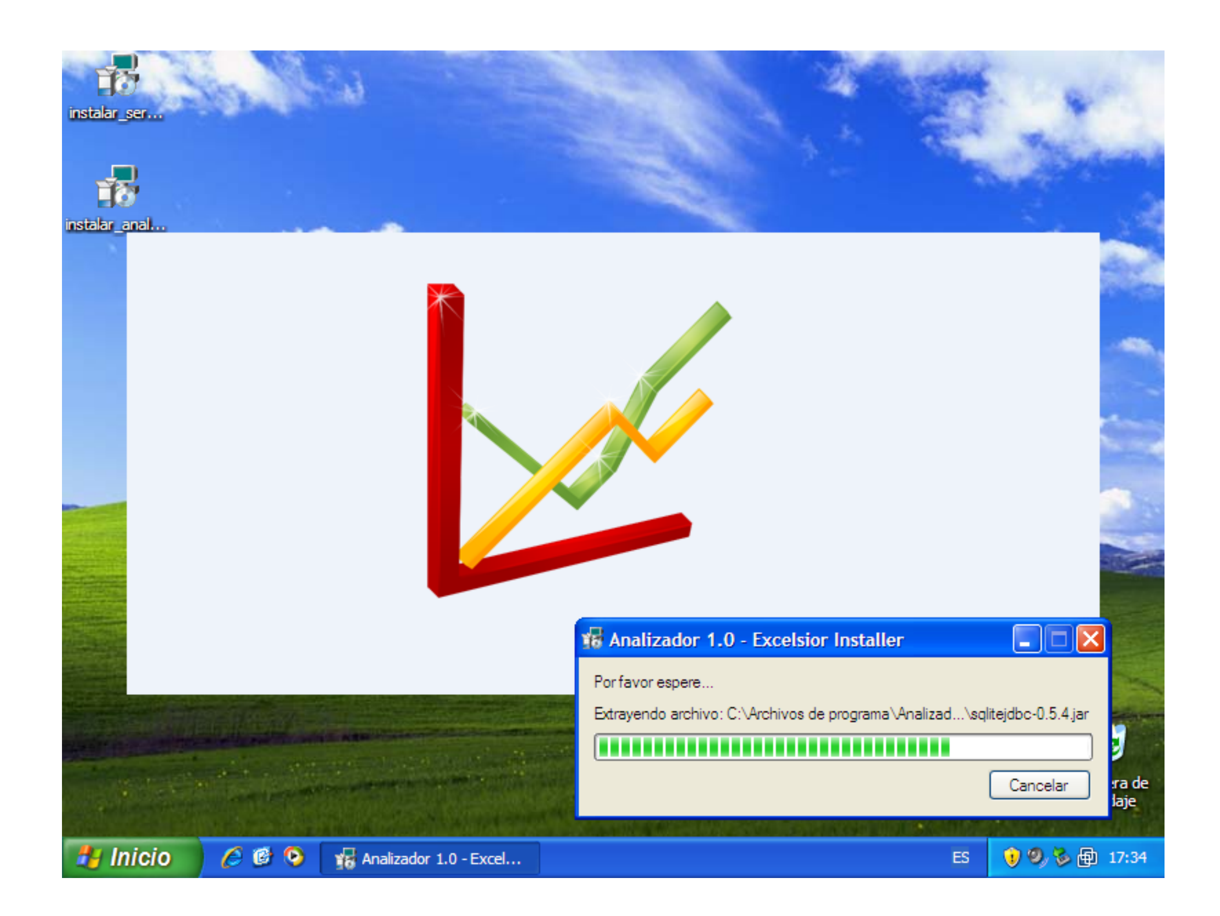


Ilustración 58– Paso de instalación de Analizador (8)

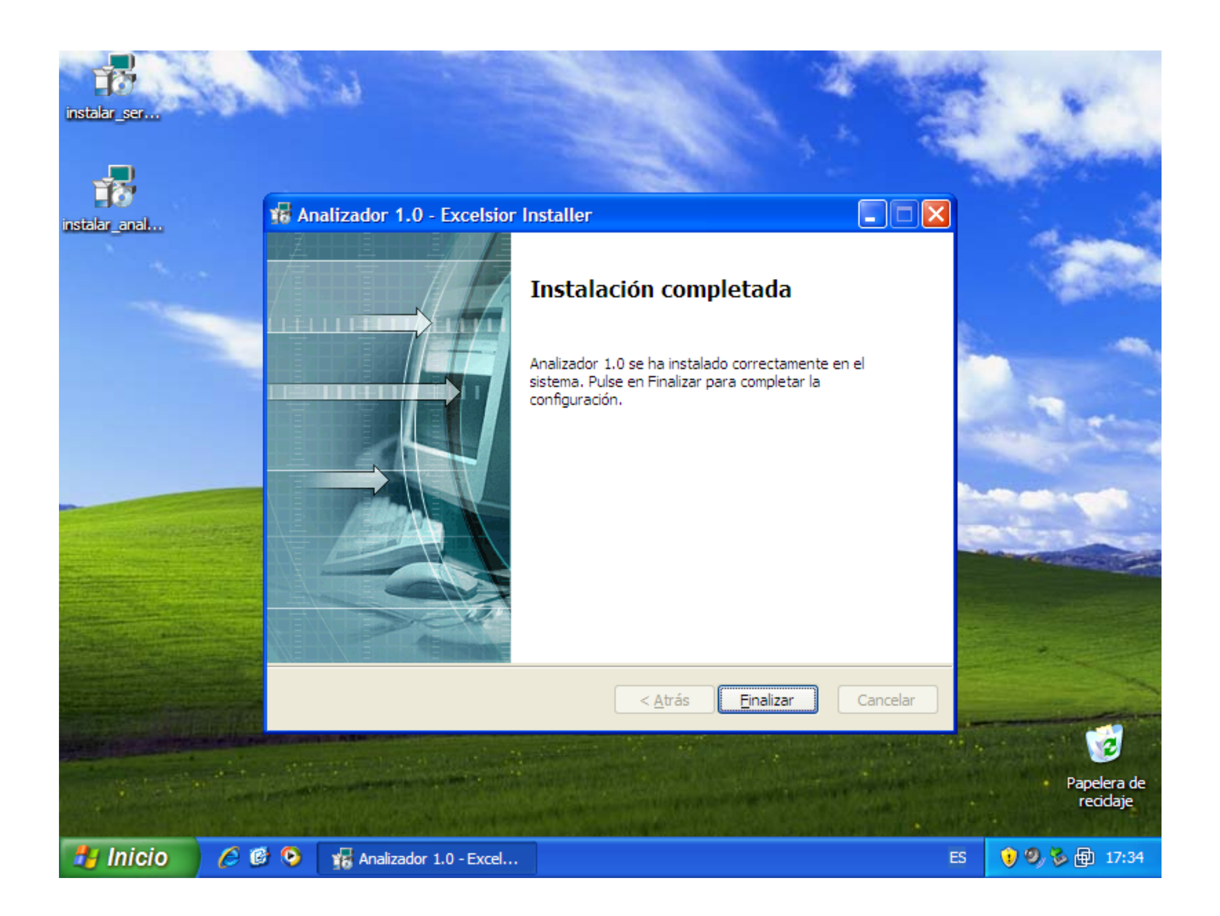


Ilustración 59– Paso de instalación de Analizador (9)

8.3. Manual de usuario de Servicio

Lo primero que hemos de hacer para ejecutar el servicio, es posicionarnos en el directorio donde se encuentran instalados los archivos correspondientes a esta aplicación.

Una vez allí, nos encontraremos una serie de ficheros que hemos de configurar o editar antes de ejecutar la aplicación.

Los ficheros principales son los que siguen:

- config.properties: Este fichero es el principal que configura la aplicación.
- estadisticas.db3: Fichero de base de datos SQLite, que por defecto, es donde se guardan las estadísticas.

- estadisticas.properties: Fichero de configuración de las estadísticas. Se aconseja no modificar.

- log4j.email.properties: Ejemplo de configuración de Log4J, para envío de mensajes de error por correo electrónico.

- log4j.properties: Fichero de Log4J que emplea Servicio y su configuración predeterminada.

- tareas.properties: Fichero de lista de tareas.

Vamos a empezar a ver el fichero principal, config.properties.

8.3.1. CONFIG.PROPERTIES

Para empezar, hay que indicar que el carácter # indica que todo lo que le sigue después de él y hasta final de línea es un comentario.

También hemos de decir que el orden de las líneas es intercambiable. El orden indicado, es el que se considera más coherente para que el usuario pueda hacer cambios.

Cada línea está formada por un par de elementos separados por el carácter =.

```
#
#      S      E      R      V      I      C      I      O
#
josejamilena.pfc.servidor.Main.tareas.txt = tareas.properties
josejamilena.pfc.servidor.conexion.Comun.log4j.properties = log4j.properties
josejamilena.pfc.servidor.conexion.Estadisticas.driver = org.sqlite.JDBC
josejamilena.pfc.servidor.conexion.Estadisticas.url = jdbc:sqlite:.\
\estadisticas.db3
josejamilena.pfc.servidor.conexion.Estadisticas.estadisticas.properties =
estadisticas.properties
josejamilena.pfc.servidor.tareas.CargarTareas.separador = ;
josejamilena.pfc.servidor.tareas.CargarTareas.SQL = SQL
josejamilena.pfc.servidor.tareas.CargarTareas.PLSQL = PLSQL
# fichero de estadisticas SQLite para distrivuir por red
josejamilena.pfc.servidor.tcp.ServidorTCP.ficherosqlite = estadisticas.db3
josejamilena.pfc.servidor.tcp.ServidorTCP.TCPPort = 3185

#
```

```
#      C      H      A      R      T      S      E      R      V      E      R
#
josejamilena.pfc.servidor.chartserver.driver = org.sqlite.JDBC
josejamilena.pfc.servidor.chartserver.url = jdbc:sqlite:.\\estadisticas.db3
josejamilena.pfc.servidor.chartserver.TCPPort = 80
```

La línea *josejamilena.pfc.servidor.Main.tareas.txt* indica el nombre del fichero donde se encuentra la programación de las tareas a ejecutar por Servicio. La descripción de su contenido la trataremos más adelante.

La línea *josejamilena.pfc.servidor.conexion.Comun.log4j.properties* indica el nombre del fichero de configuración de Apache Log4J.

En *josejamilena.pfc.servidor.conexion.Estadisticas.driver* drive JDBC para acceder al almacenamiento de datos estadísticos.

En *josejamilena.pfc.servidor.conexion.Estadisticas.url* indica la URL de acceso al sistema de almacenamiento. Su formato depende del driver JDBC cargado en la línea anterior. El formato de direcciones URL, varía según el driver y ha de ver la documentación del mismo para informarse de cómo contruirla.

En *josejamilena.pfc.servidor.conexion.Estadisticas.estadisticas.properties* se indica las instrucciones de acceso a las estadísticas. Solo tiene sentido su modificación si el driver JDBC que se carga se conecta a un sistema gestor de bases de datos que no cumpla la sintaxis SQL estándar.

En *josejamilena.pfc.servidor.tareas.CargarTareas.separador* indica el separador empleado dentro del fichero de programación de tareas para separar los parámetros que forman la instrucción de tarea.

Con *josejamilena.pfc.servidor.tareas.CargarTareas.SQL* se configura el alias de identificación para ejecutar una tarea de sentencias SQL.

Con *josejamilena.pfc.servidor.tareas.CargarTareas.PLSQL* se configura el alias de identificación para ejecutar una tarea de sentencias PL/SQL o T-SQL.

En *josejamilena.pfc.servidor.tcp.ServidorTCP.ficherosqlite* indicamos el fichero de base de datos SQLite a publicar por red. Este modificador puede publicar un fichero distinto al que se está trabajando actualmente. Lo normal es que sea el mismo que con el que se está trabajando.

Con *josejamilena.pfc.servidor.tcp.ServidorTCP.TCPPort* se indica el puerto para acceso TCP a los datos de estadísticas.

El resto de los parámetros pertenecen a la configuración de ChartServer y serán analizados posteriormente.

8.3.2. TAREAS.PROPERTIES

```
# <fichero script> ; <planificación> ; <[SQL|PLSQL]> ; <driver JDBC> ; <hostname> ;
<cadena de conexión> ; <usuario> ; <contraseña>[FIN_LINEA]4
ejemplo.sql;* * * *
*;SQL;oracle.jdbc.driver.OracleDriver;vmware;jdbc:oracle:thin:@192.168.2.17:1521:XE;orac
le;oracle[FIN_LINEA]5
ejemplo_1.sql;* * * *
*;PLSQL;oracle.jdbc.driver.OracleDriver;vmware;jdbc:oracle:thin:@192.168.2.17:1521:XE;or
acle;oracle[FIN_LINEA]6
```

Cada línea de está formada por ocho parámetros, separados por la variable *josejamilena.pfc.servidor.tareas.CargarTareas.separador* indicada en el fichero *config.properties*.

Los parámetros son los que siguen:

- <fichero script>: Nombre del fichero script que contiene el código de la tarea. Normalmente los ficheros se colocan en el mismo directorio que la aplicación, aunque pueden colocarse donde se quiera si se indica la ruta correctamente.

- <planificación>: Planificación en formato *cron* de Linux. Se comenta más adelante.

- <[SQL|PLSQL]>: Se indica con SQL si el fichero contiene sentencias SQL, o PLSQL si son proceso PL/SQL o T-SQL.

[4] Fin de línea. No presente en el fichero.

[5] Fin de línea. No presente en el fichero.

[6] Fin de línea. No presente en el fichero.

- <driver JDBC>: Driver JDBC del servidor de bases de datos al que se va a acceder.

- <hostname>: Nombre o dirección del sistema gestor de bases de datos a tratar.

- <cadena de conexión>: URL de acceso al sistema gestor de bases de datos a tratar.

- <usuario>: Usuario con el que se conecta al sistema gestor de bases de datos a tratar.

- <contraseña>: Contraseña para ese usuario.

8.3.2.1. Formato de planificación de tareas

Los patrones de son como en UNIX y está formado por cinco cadenas de caracteres separadas por espacios. Pasemos ahora a ver cada parte:

Patrón de minutos. Indica durante que minuto de la hora deben de ser lanzadas las tareas. El rango de valores va de 0 a 59.

Patrón de horas. Indica durante que hora de la hora deben de ser lanzadas las tareas. El rango de valores va de 0 a 23.

Patrón de días. Indica los días del mes durante los que son lanzadas las tareas. El rango de valores va de 1 a 31.

Patrón de meses. Indica los meses durante los que son lanzadas las tareas. El rango de valores va de 1 (Enero) a 12 (Diciembre), también admite los alias "jan", "feb", "mar", "apr", "may", "jun", "jul", "aug", "sep", "oct", "nov" y "dec".

Patrón de días de la semana. Indica los días de la semana durante los que son lanzadas las tareas. El rango de valores va de 0 (Domingo) a 6 (Sábado), también admite los alias "sun", "mon", "tue", "wed", "thu", "fri" y "sat".

Todos admiten el carácter comodín de * indicando "cada minuto de la hora", "cada hora del día", "cada día del mes", "cada mes del año" y "cada día de la semana", de acuerdo de la posición donde sean usados.

Solo se puede lanzar una tarea que tenga los cinco patrones correctamente rellenos.

Ejemplos:

5 * * * * Este patrón hace que la tarea se lance una vez cada hora en el minuto 5 de la misma (00:05, 01:05, 02:05 etc.).

**** * * * **** Este patrón indica un lanzamiento cada minuto.

**** 12 * * Mon*** Este patrón indica que se lance cada minuto durante las 12:00 de un lunes.

**** 12 16 * Mon*** Este patrón indica que se lance cada minuto durante las 12:00 de un lunes, pero solo si este lunes es el día 16 del mes.

Cada patrón puede contener dos o más valores separados por comas.

59 11 * * 1,2,3,4,5 Este patrón hace que la tarea se lance a las 11:59 los lunes, martes, miércoles, jueves y viernes.

Se pueden definir intervalos con el carácter - como separador.

59 11 * * 1-5 Este patrón es equivalente al anterior.

Se puede emplear el carácter / para indicar periodos de tiempo fraccionados de la forma a/b. Es válido si se verifica que el valor de la izquierda, dividido por el de la derecha da un resultado entero (a% b == 0).

***/15 9-17 * * *** Este patrón hace que la tarea se lance cada 15 minutos entre las 9:00 y las 17:00 (9:00, 9:15, 9:30, 9:45 y así hasta su última ejecución a las 17:45).

Todos los modificadores se pueden mezclar.

**** 12 10-16/2 * **** Este patrón hace que la tarea se lance cada minuto durante las 12:00 del día, pero solo si el día es el 10, 2th, o 16 del mes.

**** 12 1-15,17,20-25 * **** Esté patrón hace que la tarea se lance durante las 12:00 del día si es la primera quincena del mes, o del día 20th al 25, o el día 17.

También se pueden combinar patrones con el carácter |.

0 5 * * *|8 10 * * *|22 17 * * * Hace que se lance la tarea todos los días a las 05:00, 10:08 y 17:22.

8.3.3. ESTADISTICAS.PROPERTIES

Esta es la configuración por defecto. Si se emplea un driver de un sistema gestor de bases de datos que no sea SQL, se puede modificar, pero no es recomendable.

```
insertarEstadistica = INSERT INTO ESTADISTICAS VALUES (?,?,?,?,?)
iniciarEstructuraEnBD = CREATE TABLE ESTADISTICAS ( TIEMPO NUMBER NOT NULL,
FECHA NUMBER NOT NULL, TIPO VARCHAR2(30), HOST_SGBD VARCHAR2(30), HOST_CLIENTE
VARCHAR2(30) )
mostrarMedia = SELECT SUM(TIEMPO)/COUNT(TIEMPO) AS MEDIA_TIEMPO FROM
ESTADISTICAS
mostrarMediaSeleccion = SELECT SUM(TIEMPO)/COUNT(TIEMPO) AS
MEDIA_TIEMPO_SELECT FROM ESTADISTICAS WHERE TIPO = 'SELECION'
mostrarMediaManipulacionDatos = SELECT SUM(TIEMPO)/COUNT(TIEMPO) AS
MEDIA_TIEMPO_MD FROM ESTADISTICAS WHERE TIPO = 'MANIPULACION_DE_DATOS'
mostrarMediaManipulacionTablas = SELECT SUM(TIEMPO)/COUNT(TIEMPO) AS
MEDIA_TIEMPO_MT FROM ESTADISTICAS WHERE TIPO = 'MANIPULACION_DE_TABLAS'
mostrarMediaOtrasOperaciones = SELECT SUM(TIEMPO)/COUNT(TIEMPO) AS
MEDIA_TIEMPO_OTRAS FROM ESTADISTICAS WHERE TIPO = 'NO_DEFINIDO'
mostrarMediaOtrasOperacionesPrueba = SELECT SUM(TIEMPO)/COUNT(TIEMPO) AS
MEDIA_TIEMPO_MT FROM ESTADISTICAS_TMP WHERE TIPO = 'NO_DEFINIDO'
```

8.3.4. LOG4J.PROPERTIES

Define los parámetros de Apache Log4J.

Este es el fichero por defecto:

```
#log4j.rootCategory=ALL, LOGFILE, CONSOLE, EMAIL
log4j.rootCategory=ALL, LOGFILE, CONSOLE

log4j.appender.LOGFILE=org.apache.log4j.DailyRollingFileAppender
log4j.appender.LOGFILE.file=log/diario.log
log4j.appender.LOGFILE.append=true
log4j.appender.LOGFILE.DatePattern='.'yyyy-MM-dd
log4j.appender.LOGFILE.layout=org.apache.log4j.PatternLayout
log4j.appender.LOGFILE.layout.ConversionPattern=[%d{yyyy-mm-dd hh:mm},%6.6r]
%-5p %c [%t]%x - %m%n
log4j.appender.LOGFILE.Threshold=ALL
log4j.appender.LOGFILE.ImmediateFlush=true

log4j.appender.CONSOLE=org.apache.log4j.ConsoleAppender
log4j.appender.CONSOLE.layout=org.apache.log4j.PatternLayout
log4j.appender.CONSOLE.layout.ConversionPattern= =>%5p [%t] (%c)(%F:%L)
(%d{yyyy-mm-dd hh:mm}) %n      %m%n

# EMAIL
#log4j.appender.EMAIL=org.apache.log4j.net.SMTPAppender
#log4j.appender.EMAIL.SMTPHost=localhost
#log4j.appender.EMAIL.From=test@localhost
#log4j.appender.EMAIL.To=test@localhost
#log4j.appender.EMAIL.Subject=Ya mandao errores mierda es estos...
##log4j.appender.EMAIL.layout=org.apache.log4j.HTMLLayout
#log4j.appender.EMAIL.layout=org.apache.log4j.PatternLayout
#log4j.appender.EMAIL.layout.ConversionPattern=[%d] [%t] %-5p %c %x - %m%n
#log4j.appender.EMAIL.Threshold=ERROR
##log4j.appender.EMAIL.BufferSize=1
#log4j.appender.EMAIL.SMTPDebug=true
```

Este es el fichero por defecto para empleo del aviso de mensajes por correo electrónico:

```
log4j.rootCategory=ALL, LOGFILE, CONSOLE, EMAIL

log4j.appender.LOGFILE=org.apache.log4j.DailyRollingFileAppender
log4j.appender.LOGFILE.file=log/diario.log
log4j.appender.LOGFILE.append=true
log4j.appender.LOGFILE.DatePattern='.'yyyy-MM-dd
log4j.appender.LOGFILE.layout=org.apache.log4j.PatternLayout
log4j.appender.LOGFILE.layout.ConversionPattern=[%d{yyyy-mm-dd hh:mm},%6.6r]
%-5p %c [%t]%x - %m%n
log4j.appender.LOGFILE.Threshold=ALL
log4j.appender.LOGFILE.ImmediateFlush=true

log4j.appender.CONSOLE=org.apache.log4j.ConsoleAppender
log4j.appender.CONSOLE.layout=org.apache.log4j.PatternLayout
log4j.appender.CONSOLE.layout.ConversionPattern= =>%5p [%t] (%c)(%F:%L)
(%d{yyyy-mm-dd hh:mm}) %n      %m%n

# EMAIL
log4j.appender.EMAIL=org.apache.log4j.net.SMTPAppender
log4j.appender.EMAIL.SMTPHost=localhost
log4j.appender.EMAIL.From=test@localhost
log4j.appender.EMAIL.To=test@localhost
log4j.appender.EMAIL.Subject=Ya mandao errores mierda es estos...
#log4j.appender.EMAIL.layout=org.apache.log4j.HTMLLayout
```

```
log4j.appender.EMAIL.layout=org.apache.log4j.PatternLayout
log4j.appender.EMAIL.layout.ConversionPattern=[%d] [%t] %-5p %c %x - %m%n
log4j.appender.EMAIL.Threshold=ERROR
#log4j.appender.EMAIL.BufferSize=1
log4j.appender.EMAIL.SMTPDebug=true
```

Para más información de cómo configurar los ficheros de Log4J. Véase el manual en línea en http://logging.apache.org/log4j/1.2/manual.html

8.3.5. *EJECUCIÓN DE SERVICIO*

Se lanza, una vez configurado todo con la instrucción:

java –jar servicio.jar

8.4. MANUAL DE USUARIO DE CHARTSERVER

ChartServer se ejecuta como componente de Servicio y emplea su mismo fichero de configuración.

8.4.1. *CONFIG.PROPERTIES PARA CHARTSERVER*

```
#
#     C     H     A     R     T     S     E     R     V     E     R
#
josejamilena.pfc.servidor.chartserver.driver = org.sqlite.JDBC
josejamilena.pfc.servidor.chartserver.url = jdbc:sqlite:.\\estadisticas.db3
josejamilena.pfc.servidor.chartserver.TCPPort = 80
```

Este fichero es el fragmento no comentado del fichero, luego sigue las mismas directrices. Se emplea # como comentario de todo lo que le sigue. Y también en cada línea definimos pares de valores.

En la variable *josejamilena.pfc.servidor.chartserver.driver* definimos el driver usado para los datos de estadísticas a publicar. No tiene porque se los mismos que Servicio está obteniendo actualmente.

En la variable *josejamilena.pfc.servidor.chartserver.url* definimos la URL para acceder a los datos a publicar.

En la variable *josejamilena.pfc.servidor.chartserver.TCPPort* definimos el puerto donde se abre la conexión TCP/IP donde se publicará las gráficas.

8.4.2. Ejecución de ChartServer

Se lanza, una vez configurado todo con la instrucción:

java –jar josejamilena.pfc.servidor.chartserver.jar

8.4.3. Supervisión de datos desde navegador web

No conectamos al servidor de Servicio que deseamos revisar. Por ejemplo, *localhost* que emplea el puerto TCP *80*. Lo escribimos en el navegador y se nos muestra lo que sigue.

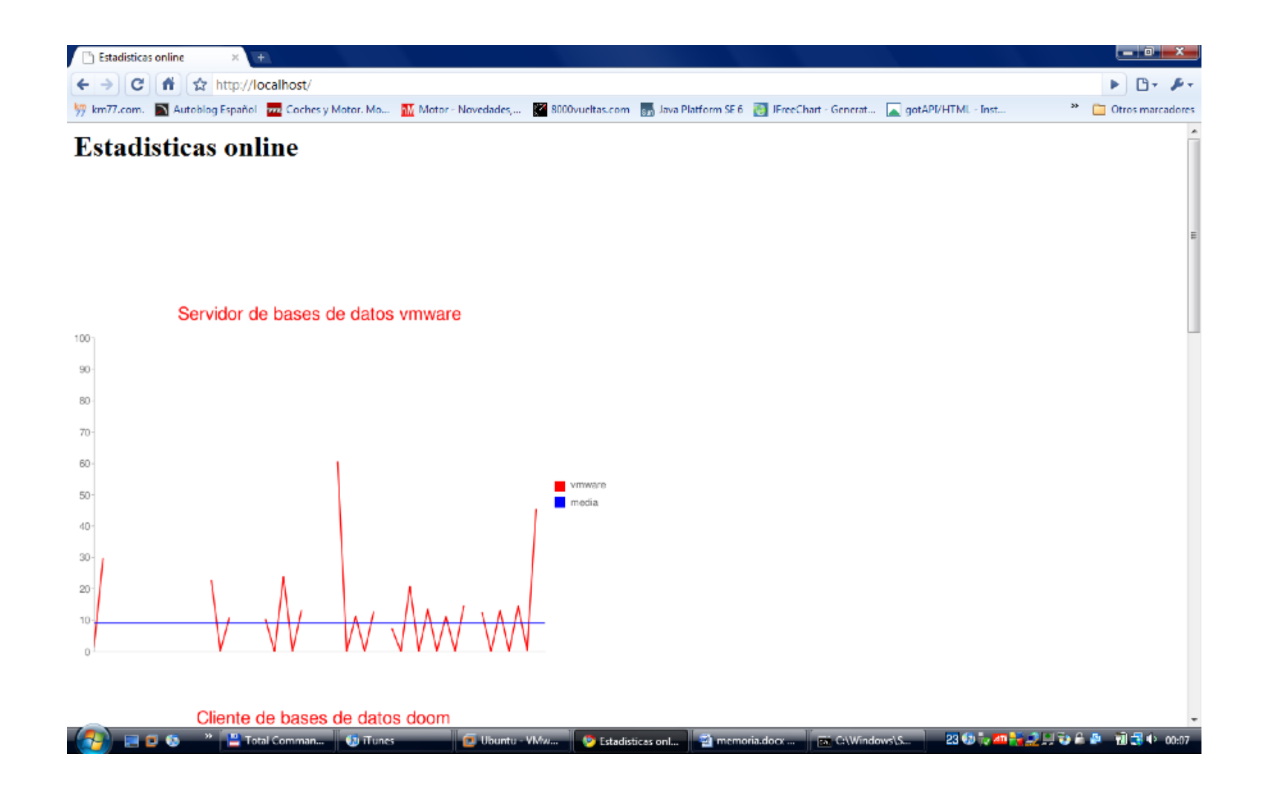


Ilustración 60 – Captura de ejemplo de uso de ChartServer

En la página se muestran distintos gráficos. En el eje de ordenadas se muestran los porcentajes de tiempo y en el de abscisas la evolución en el tiempo. Si la línea es discontinua como en el ejemplo son tiempos donde, por la programación u otros motivos, no se ha efectuado peticiones.

Las gráficas no tienen intención de que sean analíticas, sino de la supervisión de la recolección de datos.

8.5. Manual de usuario de Cliente de Estadísticas

Se lanza con la orden:

java –jar josejamilena.pfc.cliente.estadisticas.jar <host> <puerto>

Siendo:

- Host, el equipo proveedor de BD de estadísticas.
- Puerto, el puerto TCP por el que accede.

8.6. Manual de usuario de Analizador

Lo primero que vemos cuando arranca la aplicación Analizador es su característica pantalla de bienvenida.

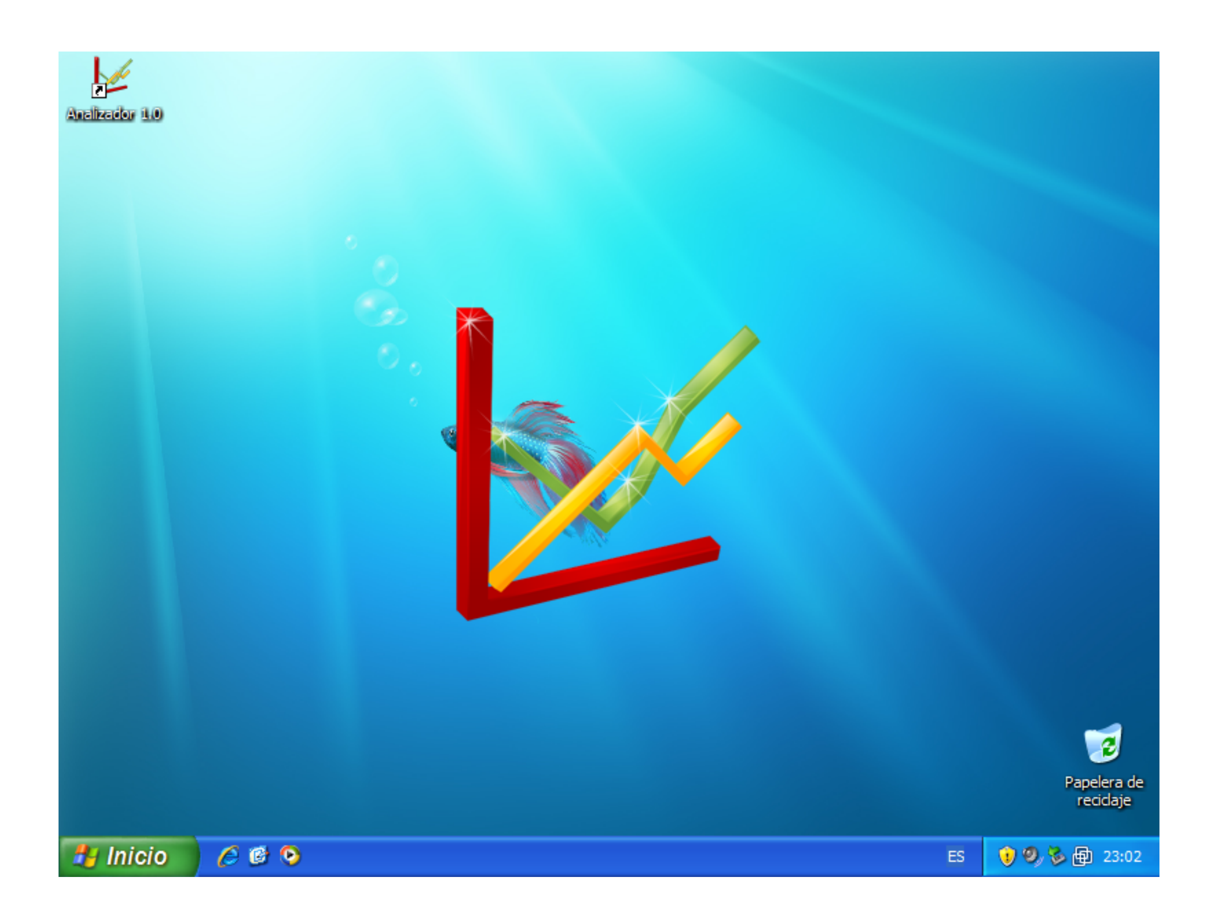


Ilustración 61 – Captura de ejemplo de uso de Analizador (1)

Tras unos instantes vemos la interfaz de la aplicación vestida con la suavizada característica del nuevo *look and feel* de la SWING de Java llamada Nimbus.

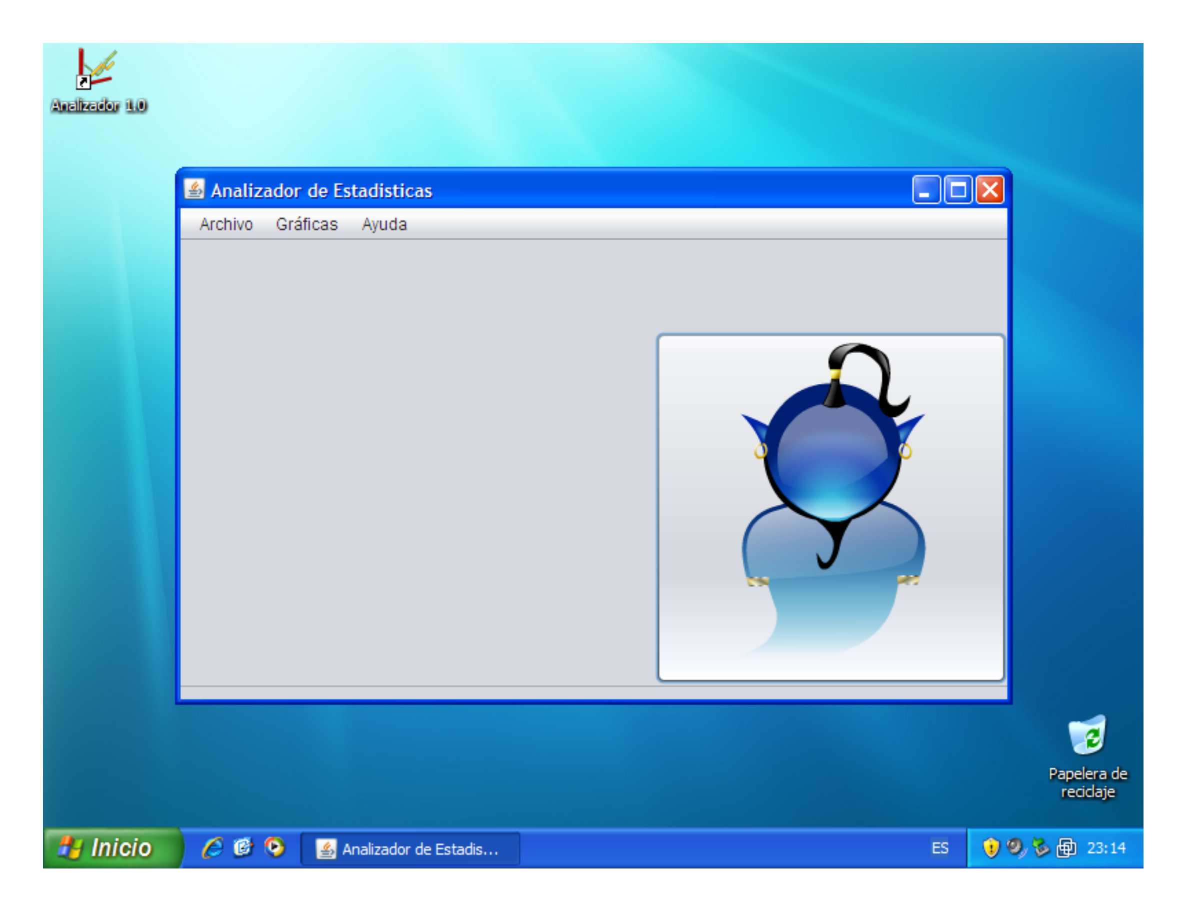


Ilustración 62 – Captura de ejemplo de uso de Analizador (2)

La interfaz tiene un menú superior que contiene el acceso a las funciones. Vamos a empezar repasando el menú Archivo.



Ilustración 63 – Captura de ejemplo de uso de Analizador (3)

El menú Archivo tiene cuatro opciones:

- Abrir fichero de estadísticas.
- Recibir fichero por red.
- Exportar como .CSV
- Salir

Abrir fichero de estadísticas. Abre un archivo SQLite dentro del equipo local. Al utilizar esta opción se nos abre un menú de exploración para buscar el archivo.

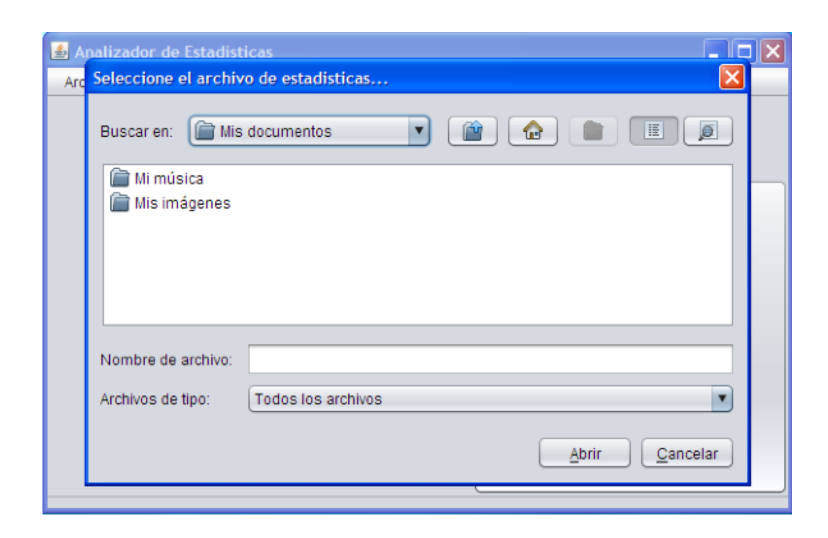


Ilustración 64 – Captura de ejemplo de uso de Analizador (4)

Si seleccionamos abrir fichero por red, nos sale un formulario donde indicamos el nombre de host del que queremos recibir los datos, y el puerto TCP para acceder al mismo. El puerto por defecto es el 3185.

Para recibir las estadísticas hacemos clic en el botón “Obtener estadísticas”, que nos creará un fichero de base de datos SQLite, en el directorio temporal del sistema operativo durante la ejecución de la aplicación, con los datos obtenidos por red.

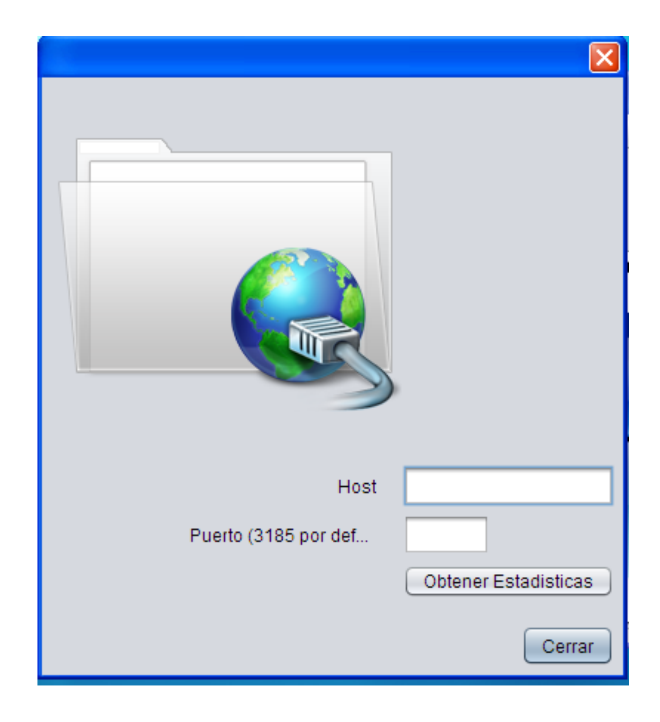


Ilustración 65 – Captura de ejemplo de uso de Analizador (5)

Si no ha habido problemas se nos muestra este mensaje

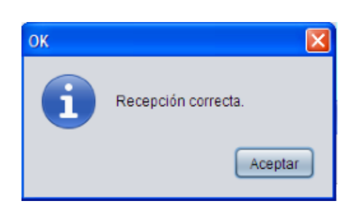


Ilustración 66 – Captura de ejemplo de uso de Analizador (6)

Otra opción que dispone es de exportar los datos con los que está trabajando actualmente Analizador en formato de valores separador por comas. Con esta opción se nos vuelve a abrir otro explorador de archivos para que indiquemos la ruta donde se almacenarán el archivo.

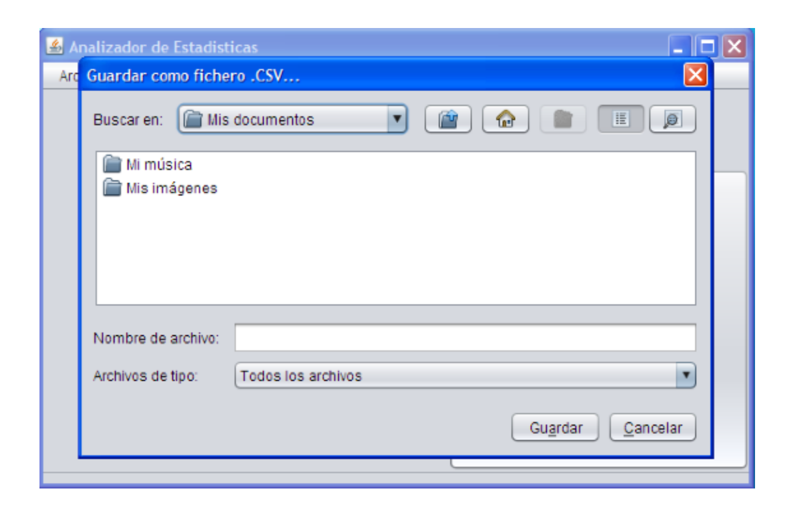


Ilustración 67 – Captura de ejemplo de uso de Analizador (7)

Pasemos ahora al menú de Gráficas. Aquí nos encontramos las opciones de:

- Gráfico por SGBD. Con esta gráfica se muestra la relación entre un sistema gestor de bases de datos con un determinado script.
- Gráfico por Script. Muestra la relación de ejecución de un cliente con un sistema gestor de bases de datos.
- Gráfico por Cliente. Muestra la relación entre un cliente con un script determinado.

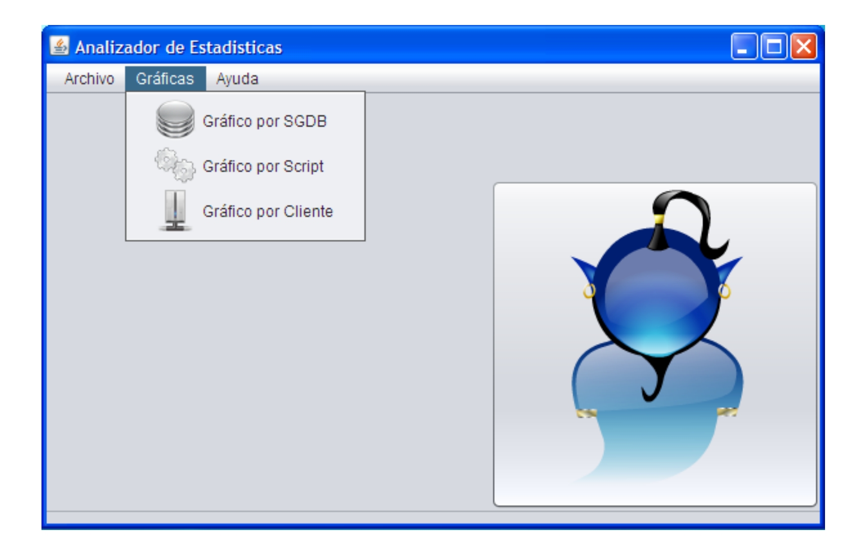


Ilustración 68 – Captura de ejemplo de uso de Analizador (8)

Las tres entradas del menú funcionan de una manera similar. En las tres se nos abre un asistente que nos ayuda a seleccionar los dos parámetros en los que nos queremos centrar en el estudio.

Para generar un “Gráfico por SGBD”, primero seleccionamos el sistema gestor de bases de datos a analizar y luego los distintos script a analizar.

Para generar un “Gráfico por Script”, primero seleccionamos el cliente y luego el correspondiente sistema gestor de bases de datos a analizar.

Para generar un “Gráfico por Cliente”, primero seleccionamos el cliente a analizar y luego los distintos script.

Vamos a ver ahora que es lo que nos encontraríamos. Todo el proceso de análisis viene más detallado en el Apéndice 2 con un ejemplo de uso.

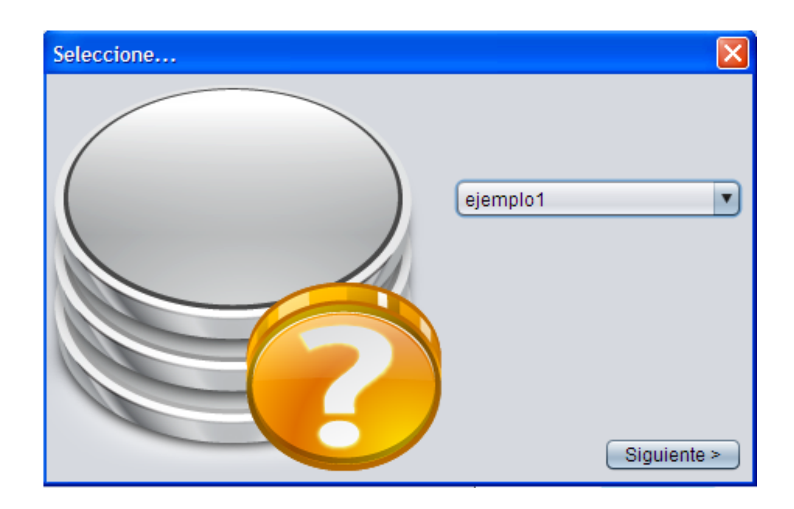


Ilustración 69 – Captura de ejemplo de uso de Analizador (9)

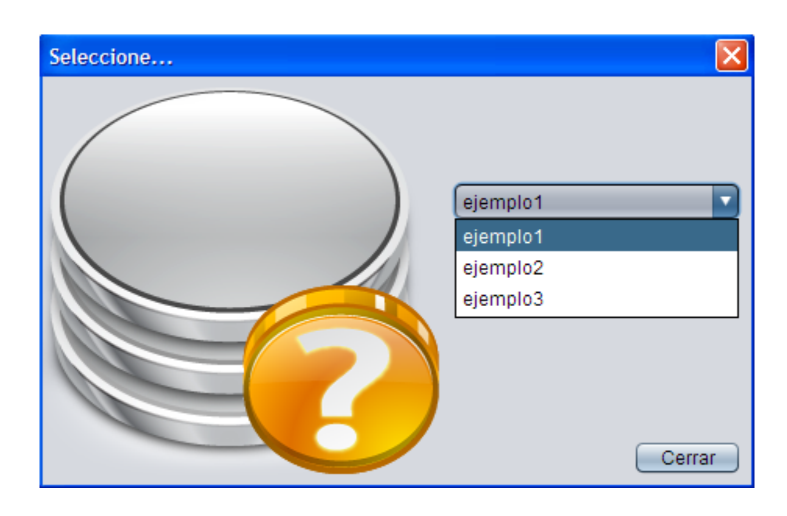


Ilustración 70 – Captura de ejemplo de uso de Analizador (10)

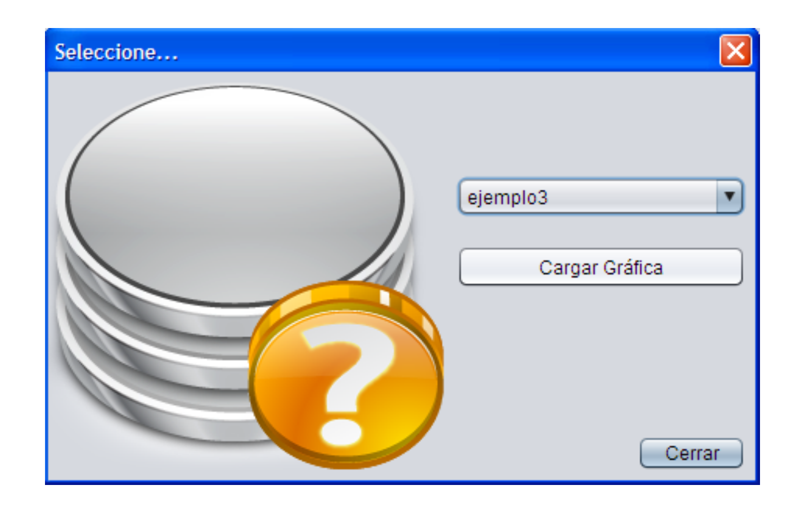


Ilustración 71 – Captura de ejemplo de uso de Analizador (11)

Tras el último paso haríamos clic en "Cargar Gráfica", y obtendríamos algo así.

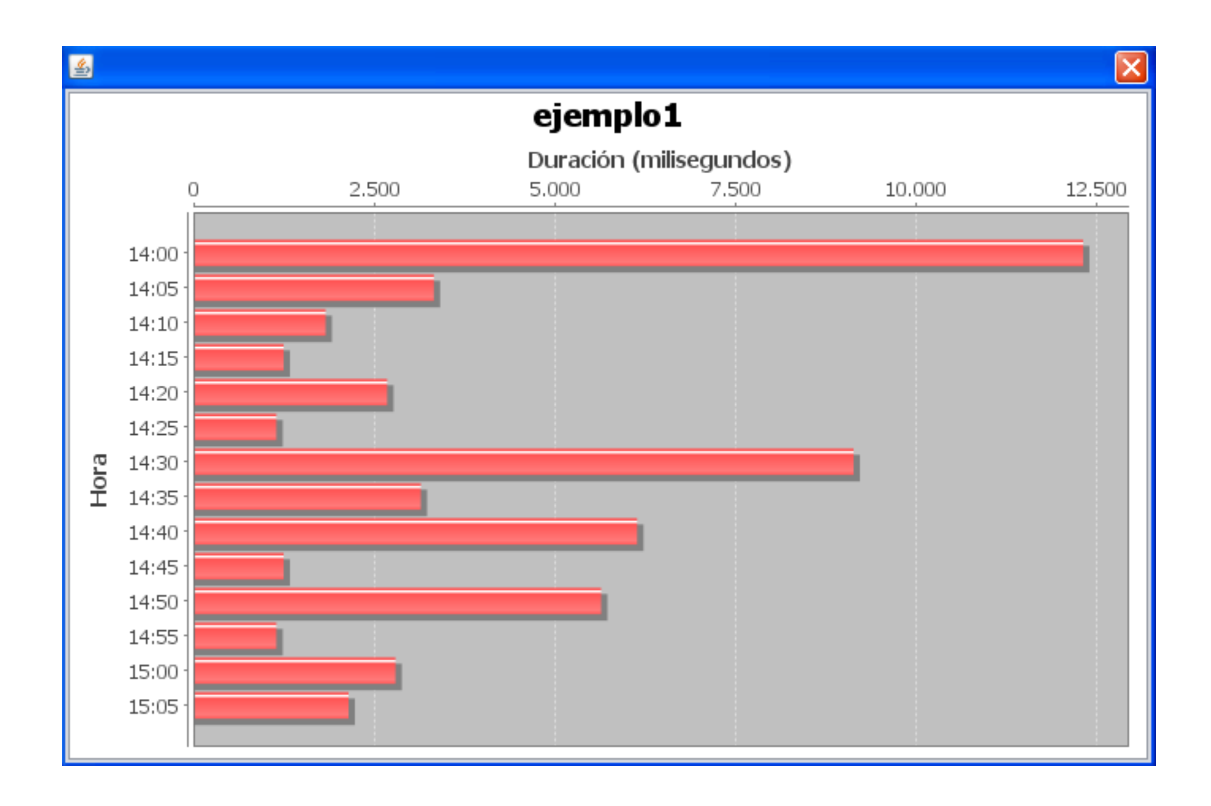


Ilustración 72 – Captura de ejemplo de uso de Analizador (12)

Nos quedaría ver el menú "Acerca de", del que obtendríamos el siguiente dialogo emergente.

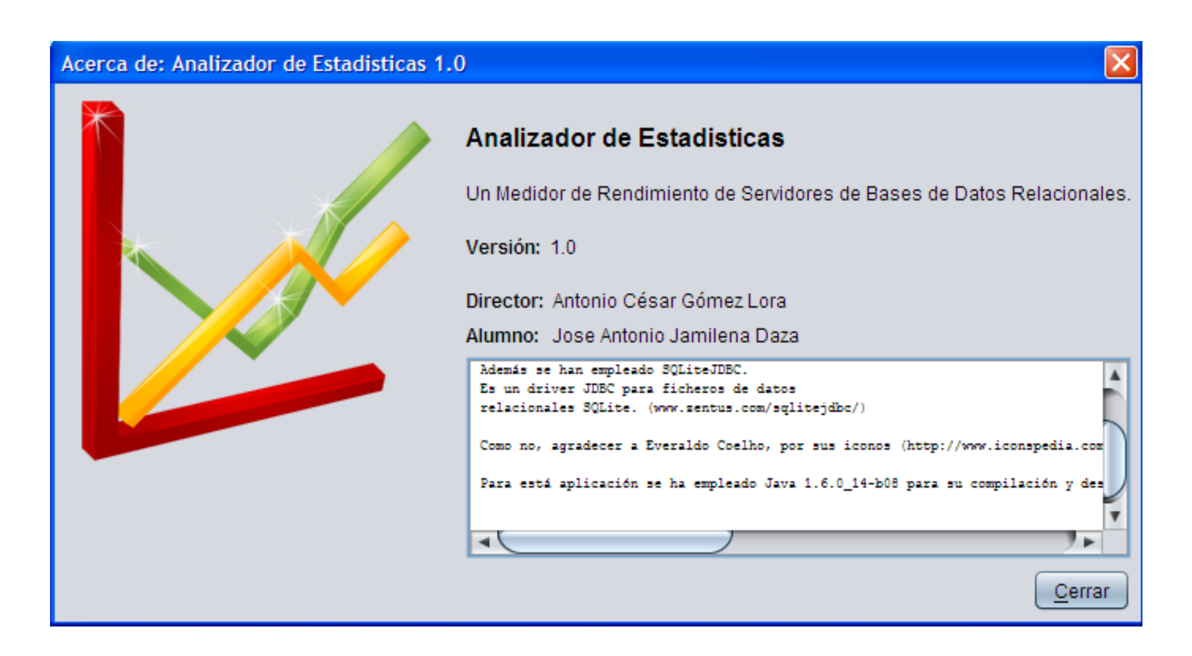


Ilustración 73 – Captura de ejemplo de uso de Analizador (13)

9. Apéndice 2: Ejemplo de uso para analizar un sistema gestor de bases de datos

9.1. Descripción del escenario de pruebas

Para las pruebas ha sido empleado VMware Workstation 6.5 ACE con dos configuraciones idénticas con estas características:

- 512 Megabytes de RAM
- 8 Gigabytes de disco duro SCSI
- Conexión en red por puente
- 1 procesador

Uno de los equipos tendrá instalado Oracle Express Edition 10g sobre un sistema operativo Windows Home Edition. El otro equipo tendrá instalado Ubuntu 9.04 Desktop Edition con la aplicación Servicio lanzando consultas.

Las máquinas virtuales correrán sobre un portátil Toshiba con Windows Vista Home Premium con 2 Gigabytes, y en esta misma máquina es donde ejecutaremos la aplicación Analizador para realizar el estudio de los datos obtenidos.

Para empezar vamos a comentar los scripts que han sido empleados para efectuar las pruebas. Van a usarse dos script. Uno SQL estándar que contiene inserciones actualizaciones y creación y borrado de datos, comentar como curiosidad que los datos empleados fueron obtenidos de las tablas publicadas en http://www.top500.org. Y otro

de PL/SQL, con creación y ejecución de procesos simples. Los listados de código vienen incluidos en el CD-ROM.

Las ejecuciones de SQL se efectuaran cada minuto, mientras que las de procesos PL/SQL se ejecutarán cada cinco. Y durante todo el proceso y mediante el cliente web de Oracle Express, se efectuarán consultas simples para intentar someter a situaciones de estrés al sistema. He incluso se ha llevado a cabo la modificación de uno de los scripts mientras se ejecuta la aplicación Servicio.

Las dos máquinas virtuales durante la ejecución han llevado entornos limpios, recién instalados y sin nada no necesario.

La prueba se ha efectuado durante cuarenta y siete minutos.

Lo primero que vamos a hacer es generar un "Gráfico por SGBD", donde seleccionamos *vmware* como sistema gestor de bases de datos, y primero seleccionamos el script de sentencias SQL.

En las comparativas vamos a ver la diferencia de ejecución entre script SQL y procedimientos PL/SQL.

La gráfica de script SQL, podemos ver dos picos. El primero coincide con la primera ejecución del script PL/SQL, que el gestor de bases de datos parece que le da más prioridad de ejecución que las sentencias de SQL.

El otro pico se debe a hacer una prueba de forzado de fallos para ver el comportamiento frente a errores. Lo que se hizo en esos instantes fue apagar la máquina virtual forzadamente. El retraso producido en la respuesta no está causado por métricas. Sino por el control que tiene la aplicación Servicio para evitar que sus tareas se queden suspendidas o muertas por no acabar la ejecución, la ejecución de dicha tarea es suspendida.

Tras esto se procede a para la aplicación Servicio.

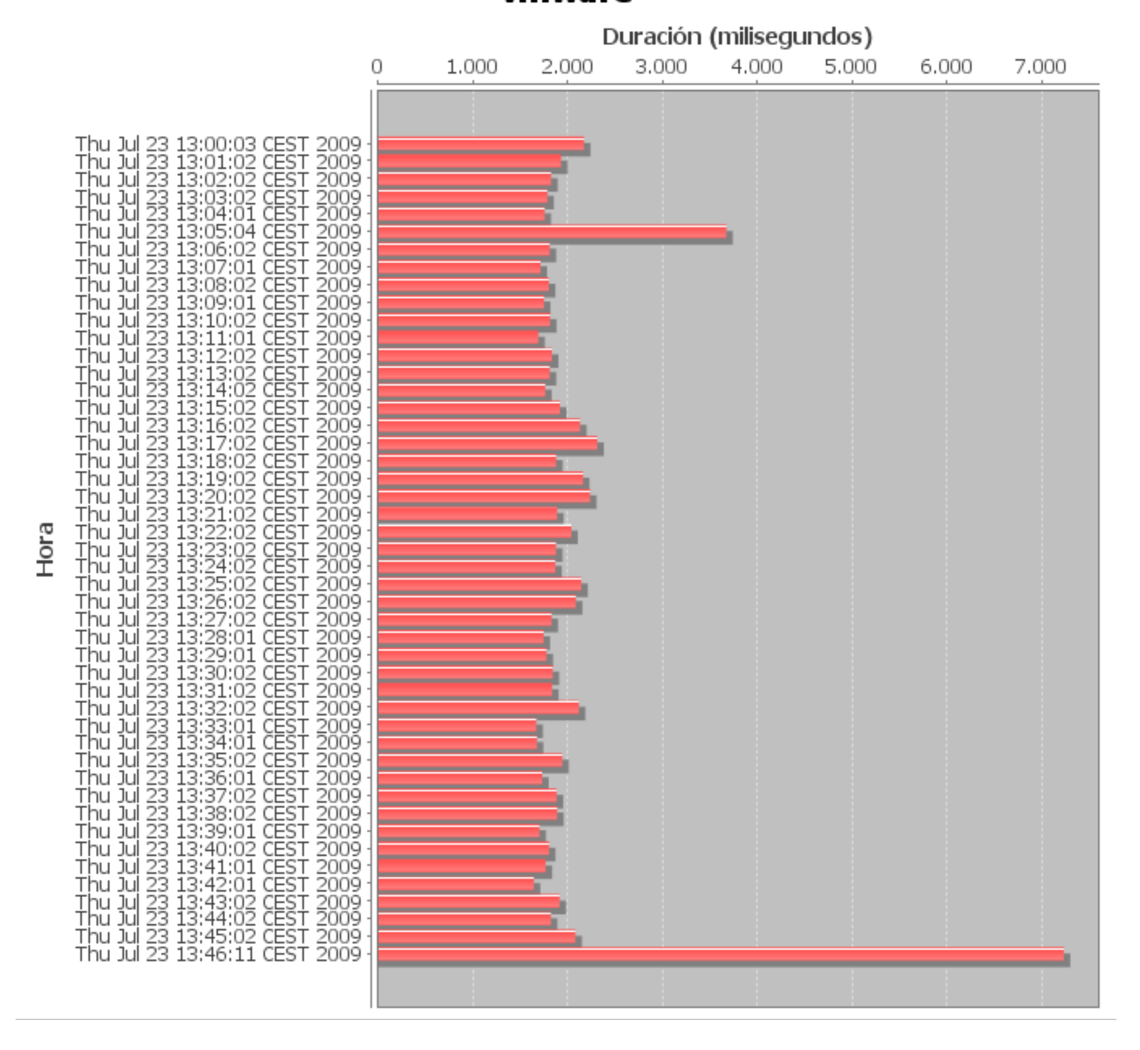


Ilustración 74 – Captura de gráfica (1)

Y ahora el de procedimientos PL/SQL. He de recordar que esta tarea se lanza cada cinco minutos, mientras que la anterior se lanza cada minuto.

Para empezar vemos que la primera ejecución del lote de procedimientos PL/SQL, es más larga que las demás ya que contiene procesos largos. Más tarde los dos picos en las medidas, son los cambios que se han indicado que se iban a hacer durante la ejecución de las tareas. Con esto podemos comprobar el funcionamiento de la caché de los sistemas gestores de bases de datos.

Cuando la ejecución es de algo nuevo tarda más que cuando se vuelven a ejecutar algo reciente, donde actúa la caché del gestor para optimizar los tiempos.

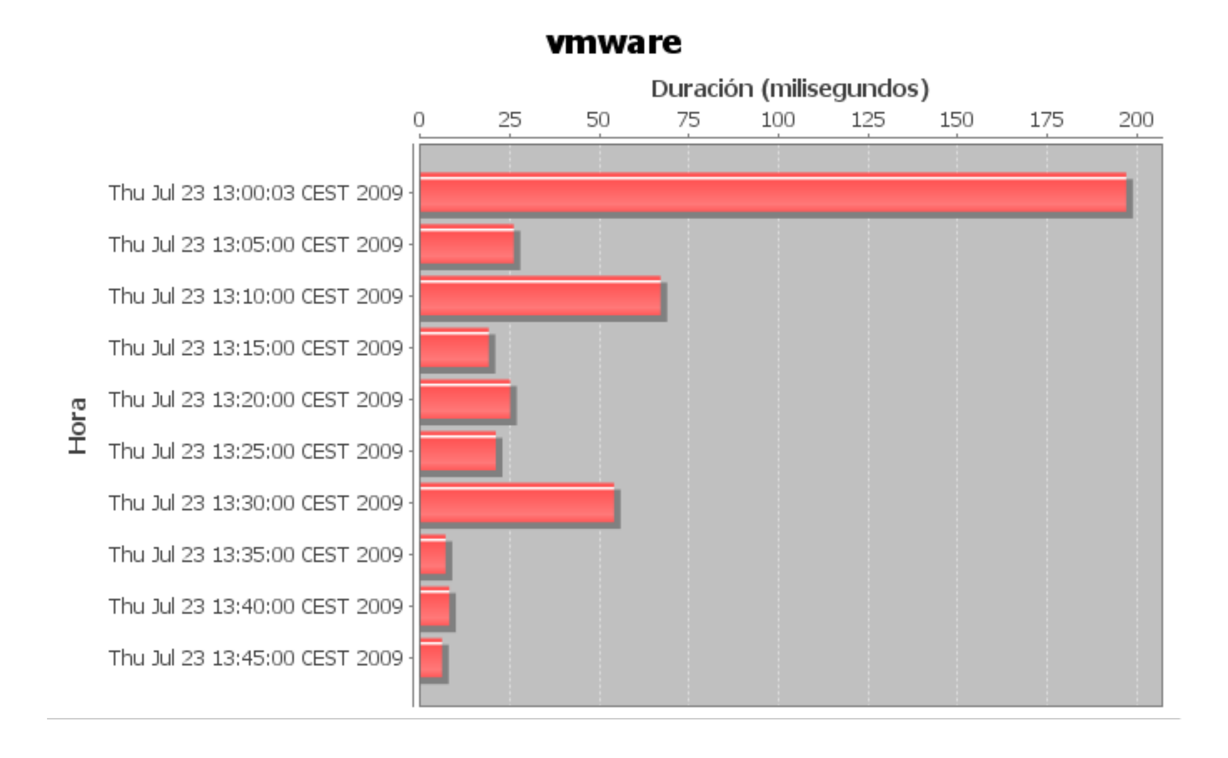
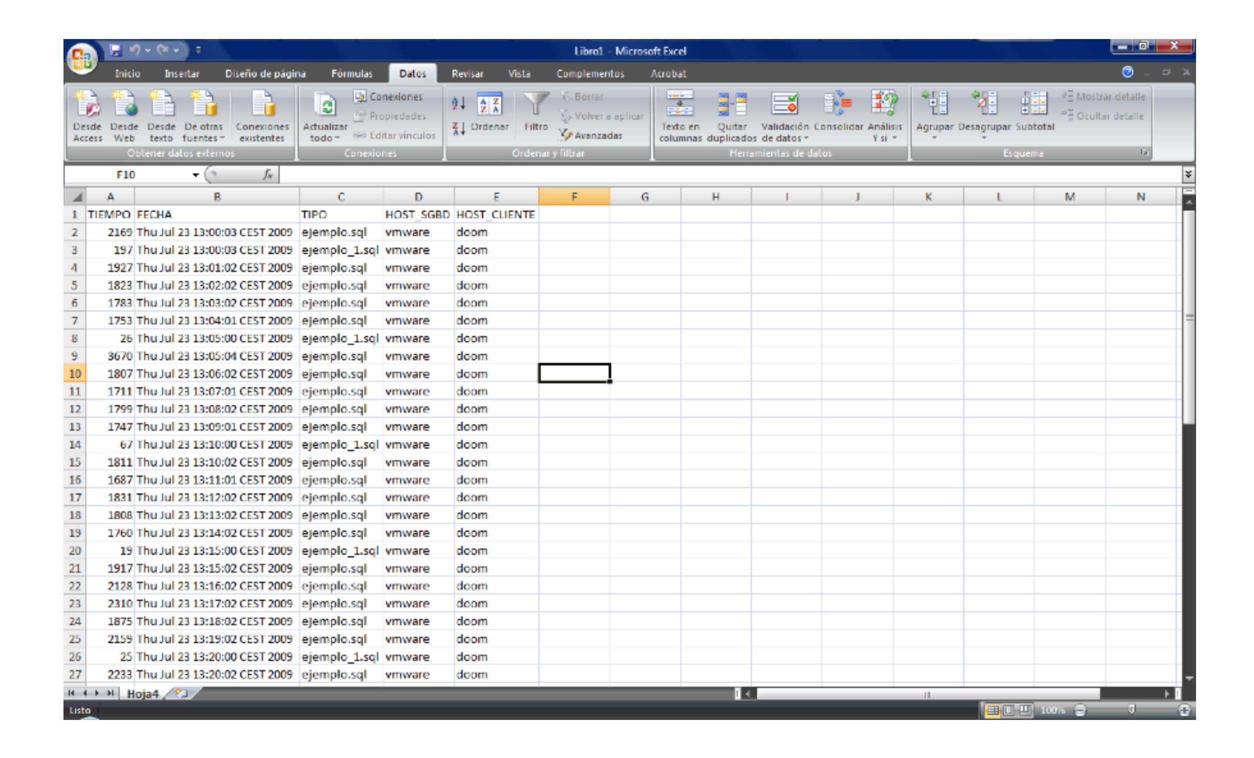


Ilustración 75– Captura de gráfica (2)

Vemos como en los datos recolectados, las diferencias entre métricas son mínimas. En cambio, cuando algo va mal, las gráficas producen picos muy grandes, o por lo menos destacados.

Si los picos se producen constantemente, hemos de echar mano de las herramientas de análisis del propio sistema gestor de bases de datos, o de las herramientas de tráfico de red, para cual puede ser el origen del fallo.

También podemos exportar los datos como archivo de valores separados por comas, importar dichos datos una hoja de cálculo, como Microsoft Excel, un software estadístico como SPSS o una herramienta como Matlab y emplear sus herramientas estadísticas para procesar los datos.



TIEMPO	FECHA	TIPO	HOST_SGBD	HOST_CLIENTE
2169	Thu Jul 23 13:00:03 CEST 2009	ejemplo.sql	vmware	doom
197	Thu Jul 23 13:00:03 CEST 2009	ejemplo_1.sql	vmware	doom
1927	Thu Jul 23 13:01:02 CEST 2009	ejemplo.sql	vmware	doom
1823	Thu Jul 23 13:02:02 CEST 2009	ejemplo.sql	vmware	doom
1783	Thu Jul 23 13:03:02 CEST 2009	ejemplo.sql	vmware	doom
1753	Thu Jul 23 13:04:01 CEST 2009	ejemplo.sql	vmware	doom
26	Thu Jul 23 13:05:00 CEST 2009	ejemplo_1.sql	vmware	doom
3670	Thu Jul 23 13:05:04 CEST 2009	ejemplo.sql	vmware	doom
1807	Thu Jul 23 13:06:02 CEST 2009	ejemplo.sql	vmware	doom
1711	Thu Jul 23 13:07:01 CEST 2009	ejemplo.sql	vmware	doom
1799	Thu Jul 23 13:08:02 CEST 2009	ejemplo.sql	vmware	doom
1747	Thu Jul 23 13:09:01 CEST 2009	ejemplo.sql	vmware	doom
67	Thu Jul 23 13:10:00 CEST 2009	ejemplo_1.sql	vmware	doom
1811	Thu Jul 23 13:10:02 CEST 2009	ejemplo.sql	vmware	doom
1687	Thu Jul 23 13:11:01 CEST 2009	ejemplo.sql	vmware	doom
1831	Thu Jul 23 13:12:02 CEST 2009	ejemplo.sql	vmware	doom
1808	Thu Jul 23 13:13:02 CEST 2009	ejemplo.sql	vmware	doom
1760	Thu Jul 23 13:14:02 CEST 2009	ejemplo.sql	vmware	doom
19	Thu Jul 23 13:15:00 CEST 2009	ejemplo_1.sql	vmware	doom
1917	Thu Jul 23 13:15:02 CEST 2009	ejemplo.sql	vmware	doom
2128	Thu Jul 23 13:16:02 CEST 2009	ejemplo.sql	vmware	doom
2310	Thu Jul 23 13:17:02 CEST 2009	ejemplo.sql	vmware	doom
1875	Thu Jul 23 13:18:02 CEST 2009	ejemplo.sql	vmware	doom
2159	Thu Jul 23 13:19:02 CEST 2009	ejemplo.sql	vmware	doom
25	Thu Jul 23 13:20:00 CEST 2009	ejemplo_1.sql	vmware	doom
2233	Thu Jul 23 13:20:02 CEST 2009	ejemplo.sql	vmware	doom

Ilustración 76– Captura de gráfica (3)

10.BIBLIOGRAFÍA

[1] Wikipedia. Iterative and incremental development. [En línea] http://en.wikipedia.org/wiki/Iterative_development.

[2] Sun Microsystems. Java™ Platform, Standard Edition 6. [En línea] http://java.sun.com/javase/6/docs/api/.

[3] Sun Microsystem. NetBeans. [En línea] http://www.netbeans.org/.

[4] Visualsvn. Visualsvn. [En línea] http://www.visualsvn.com/server/.

[5] TortoiseSVN. TortoiseSVN The coolest Interface to (Sub)Version Control. [En línea] http://tortoisesvn.net/.

[6] Sparx Systems. Enterprise Architect - UML for Business, Software and Systems. [En línea] http://www.sparxsystems.com.au/.

[7] Sun Microsystems. MySQL - The world's most popular open source database. [En línea] http://www.mysql.com/.

[8] SQLite. SQLite - Small. Fast. Reliable. [En línea] http://www.sqlite.org/.

[9] SQLiteJDBC. SQLiteJDBC. [En línea] www.zentus.com/sqlitejdbc/.

[10] Osenxpsuite. SQLite2009 Pro Enterprise Manager. [En línea] http://link.osenxpsuite.net/?uid=homepage&id=sqlite2009pro.zip.

[11] Apache Software Foundation. Logging Services. [En línea] 1999-2007. http://logging.apache.org/log4j/1.2/index.html.

[12] JFree.org. JFreeChart. [En línea] http://www.jfree.org/jfreechart/.

[13] Serena. OpenProj is a free, open source project management solution. [En línea] http://openproj.org/openproj.

[14] Sun Microsystems. OpenOffice - the free and open productivity suite. [En línea] http://www.openoffice.org/.

[15] Microsoft Corporation. Microsoft Office Visio 2007. [En línea] http://office.microsoft.com/es-es/visio/FX100487863082.aspx.

[16] Dr. Bert Scalzo, Claudia Fernandez, Donald K. Burleson, Mike Ault, Kevin Kline. Database Benchmarking: Practical Methods for Oracle & SQL Server. s.l. : Rampant TechPress, 2007. ISBN 0977671534, 9780977671533.

[17] Darwin, Ian F. Java cookbook. s.l. : O'Reilly, 2001. ISBN 0596001703, 9780596001704.